1. Publicação nº	2. Versão	3. Data	5. Distribuição
INPE-3934-RTR/087		*Junho, 1986*	☐ Interna ☐ Externa ☒ Restrita
4. Origem: *DCA/DEA*	Programa: *SISDIG/SISMAG*		

6. Palavras chaves - selecionadas pelo(s) autor(es)

MICROPROGRAMAÇÃO - COMPUTADOR MICROPROGRAMADO

TÉCNICAS DE MICROPROGRAMAÇÃO - UNIDADE CENTRAL DE PROCESSAMENTO

7. C.D.U.: *681.326.32*

8. Título *INPE-3934-RTR/087*

MANUAL DE PROGRAMAÇÃO DO COMPUTADOR ASTROP (VERSÃO I)

10. Páginas: *249*

11. Última página: *b.15*

12. Revisada por

Alderico R. de Paula Jr.

9. Autoria

Almir Cavalcanti Lemos Filho

13. Autorizada por

Marco Antonio Raupp
Diretor Geral

Assinatura responsável p/

14. Resumo/Notas

Este manual apresenta, em uma primeira versão, as informações necessárias para o desenvolvimento de "software" para o computador ASTROP. Inclui detalhes relevantes da arquitetura do computador e conjuntos de instruções referentes às versões 1.0 e 1.1 do micro programa residente na UCP/ASTROP. O tempo de execução destas instruções está no Apêndice A.

15. Observações

ABSTRACT

This manual presents, in a first version, the information necessary to the development of the software for the computer ASTROP. Important details of the computer architecture and the instruction set concerning to the versions 1.0 and 1.1 of the microprogram resident in the UCP/ASTROP are included. The execution time of these instructions is presented in the Appendix A.

SUMÁRIO

Pág.

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

Este manual apresenta, em uma primeira versão (versão 1), os aspectos necessários ao desenvolvimento de "software" para o ASTROP, que é um computador desenvolvido no âmbito do Projeto SISMAG pelo Departamento de Engenharia de Computação em Aplicações Espaciais do INPE (INPE/DCA).

Esta versão I está diretamente relacionada com as versões 1.0 e 1.1 do microprograma residente na Unidade Central de Processamento do computador ASTROP (UCP/ASTROP). A indicação de que alguma informação contida neste manual diz respeito apenas à versão 1.1 do microprograma é feita com o símbolo & entre parênteses: (&). Caso contrário a informação vale para estas duas versões do microprograma.

No Capítulo 2 são apresentadas as informações sobre o computador ASTROP que são relevantes do ponto de vista de programação. Devido à variedade de modos de endereçamento, um capítulo inteiro (Capítulo 3) é dedicado a este assunto. O Capítulo 4 detalha o conjunto de instruções (versões 1.0 e 1.1 do microprograma), enquanto o capítulo seguinte apresenta algumas técnicas de programação que usam eficientemente este conjunto de instruções. No Apêndice A estão os tempos de execução das instruções, e no Apêndice B um resumo das instruções.

Maiores detalhes sobre o barramento BASIS, a memória, os controladores de periféricos, os microprogramas, o painel, configurações, a operação e a manutenção do computador ASTROP encontram-se em manuais à parte a serem publicados posteriormente.

Neste manual:

a) os números em representação hexadecimal são escritos entre aspas e imediatamente seguidos de uma letra H maiúscula: "F35A"H;

b) a numeração dos bits que compõem "bytes", palavras e palavras longas é sempre feita com números inteiros seqüenciais com o bit menos significativo recebendo o número 0 (bit 0) e o bit mais significativo recebendo o número 7 (bit 7) no caso de "byte", o número 15 (bit 15) no caso de palavra, ou o número 31 (bit 31) no caso de palavra longa.

CAPÍTULO 2

ARQUITETURA DO SISTEMA

Neste capítulo são abordados os aspectos da arquitetura do computador ASTROP que são relevantes do ponto de vista de programação.

2.1 - BASIS

O BASIS (BArramento do SIStema) do computador ASTROP provê o único meio de comunicação:

a) da UCP/ASTROP com a memória e os controladores de periféricos;

b) para o acesso direto à memória ou controlador de periférico (ADMP) por outro controlador de periférico.

A Figura 2.1 apresenta os tipos de unidades funcionais que podem ser conectadas ao BASIS. As transferências de dados são assíncronas, envolvendo um protocolo de comunicação do tipo mestre/ escravo ("master/slave"): para todo sinal emitido pelo mestre deve haver uma resposta adequada do escravo. Portanto, a velocidade de transferência de dados não é fixa, mas depende do tempo de resposta de cada unidade escrava em particular.

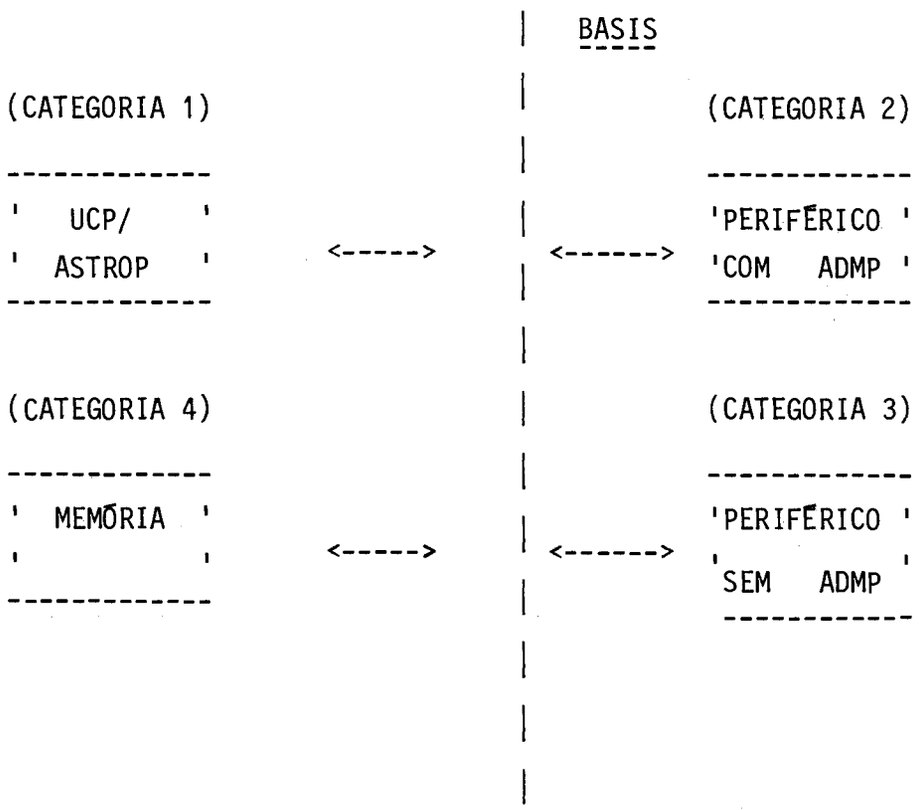


Fig. 2.1 - Tipos de unidades funcionais que podem ser conectadas ao BASIS.

De acordo com a capacidade de controle sobre o BASIS, as unidades funcionais a ele conectadas podem ser classificadas em quatro categorias:

1) UCP/ASTROP - É a unidade funcional de maior controle sobre o BASIS, agindo sempre como mestre do barramento. Normalmente, a UCP/ASTROP detém o controle do BASIS, mas pode cedê-lo, quando solicitada e segundo sua conveniência, para uma unidade funcional de categoria 2.

2) Periféricos com ADMP - Depois da UCP/ASTROP, é a unidade funcional que tem maior controle sobre o BASIS. Age como escrava nas transações que envolvem a UCP/ASTROP, mas tem a capacidade de solicitar dela o controle do barramento, para, como mestre, efetuar uma operação de acesso direto à memória ou outro controlador de periférico. Pode também interromper o processamento da UCP/ASTROP.

3) Periféricos sem ADMP - É o escravo nas transações com as unidades funcionais de categorias 1 e 2. Não tem a possibilidade de obter o controle do BASIS e agir como mestre do barramento, mas pode interromper o processamento da UCP/ASTROP.

4) Memória - É a unidade funcional com o menor controle sobre o BASIS. Age sempre como escrava em qualquer transação e não tem capacidade de interromper o processamento da UCP/ASTROP.

A forma de comunicação é a mesma para todas as unidades conectadas ao BASIS. O conjunto de sinais utilizados pela UCP/ASTROP para se comunicar com a memória e os controladores de periféricos também é utilizado nas operações de ADMP pelas unidades de categoria 2.

A entrada e saída ("I/O") no BASIS é do tipo mapeada na memória ("mappead I/O"), o que exige que todos os registros dos controladores de periféricos tenham endereço de posição de memória (Figura 2.2). Esta característica permite que as instruções do ASTROP que fazem acesso à memória também possam ser utilizadas no acesso aos dados contidos nos registros dos controladores de periféricos conectadas ao BASIS. Assim, os registros destes controladores são tratados com a mesma flexibilidade que as posições de memória.

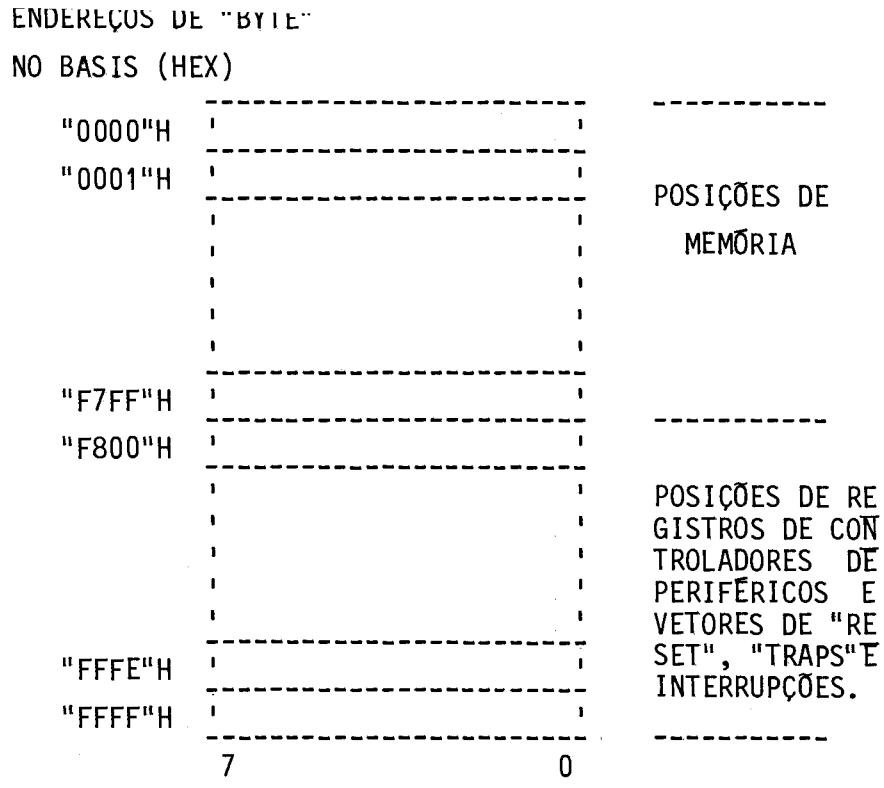


Fig. 2.2 - Alocação da capacidade de endereçamento do BASIS.

Existem 16 níveis de prioridade no BASIS para interrupção do processamento da UCP por unidades de categoria 2 e 3. Existe também um nível de prioridade para requisição de ADMP por unidade de categoria 2. No caso de haver mais de uma requisição de ADMP ao mesmo tempo, a unidade eletricamente mais próxima da UCP/ASTROP ("daisy chain") terá prioridade no atendimento.

A transferência de um bloco de informação de uma unidade de categoria 2 para a memória ou outro periférico, e vice-versa, é feita palavra a palavra (ou "byte" a "byte"), através de roubos de ciclos ("cycle stealing") da UCP/ASTROP, no uso do BASIS.

Os sinais do BASIS (Tabela 2.1) podem ser agrupados em quatro secções:

1) Secção de transferência de dados. Ao todo, 37 linhas de sinais são usadas na transferência de dados entre a UCP/ASTROP e a memória e os controladores de periféricos, ou entre um periférico de categoria 2 ou outro periférico (ADMP). O mestre do BASIS (UCP/ASTROP ou unidade funcional de categoria 2) controla a transferência de dados de uma unidade funcional para outra, que é o escravo.

2) Secção de interrupção. Fazem parte desta secção 21 linhas de sinais, relacionadas com o procedimento de interrupção do processamento da UCP/ASTROP, por unidades funcionais de categorias 2 e 3. A linha de interrupção de maior prioridade não é mascarável.

3) Secção de transferência do controle do barramento. Para uma unidade periférica de categoria 2 realizar uma operação de ADMP ela tem primeiro de obter o controle (tornar-se mestre) do BASIS. Normalmente o controle do BASIS está com a UCP/ASTROP. Entretanto, quando ela não necessita do BASIS para transferência de informação e lhe é solicitado por alguma unidade funcional de categoria 2, a UCP/ASTROP pode ceder este controle para que uma operação de ADMP se realize. Na determinação de quem é o mestre do BASIS e na transferência desta condição da UCP/ASTROP para as unidades funcionais de categoria 2 estão envolvidas quatro linhas de sinal.

4) Secção de "reset". Composta de um único sinal que se origina na UCP/ASTROP e que avisa a todas as outras unidades funcionais conectadas ao BASIS que os seus circuitos devem ser inicializados. Isto ocorre sempre que os circuitos da UCP/ASTROP são energizados, ou durante a execução da instrução RESET pelo ASTROP, ou ainda por acionamento de um "push-botton" específico existente no painel do computador ASTROP.

TABELA 2.1

SINAIS DO BASIS

NOME	MNEMÔNICO	Nº LINHAS	FUNÇÃO
1) Secção de Transferência de Dados			
Endereço	E	16	Selecionar posição de Mem. ou Registro de Periférico
Dado	D	16	Transferir informações
Leitura/Escrita	L/E	1	} Designar o sentido e o tipo de transferência de informação
Palavra/"Byte"	PAL/BTE	1	
Requer Operação	REQOP	1	} Sincronizar a transferência de informações
Permite Operação	PEROP	1	
Erro de Paridade	PARERR	1	Indicar se houve erro de paridade
	Total:	37	
2) Secção de Interrupção			
Pede interrupção	PI	16	Solicitar interrupção
Permite a Interrupção	PERIT	5	Avisar a aceitação da interrupção
	Total:	21	
3) Secção de Transferência do Controle do Barramento			
Barramento livre	BLIV	1	Indicar liberação do barramento
Requer ADMP	READM	1	} Controlar a transferência do barramento
Permissão ADMP	PEADM	1	
Fim de ADMP	FICAM	1	
	Total:	4	
4) Secção de "Reset"			
Inicie	INI	1	"Reset" para os periféricos
	Total:	1	

2.2 - UCP/ASTROP

A UCP/ASTROP é a unidade funcional que controla a alocação do BASIS para as unidades controladoras de periféricos, além de decodificar e executar as instruções. Ela apresenta as seguintes características gerais:

- Arquitetura organizada, eficiente e flexível implementada basicamente com componentes da série 74 (S e LS), componentes "bit-slice" das famílias AM2900 da Advanced Micro Devices e 9400 da Fairchild, e memórias PROMs bipolares.

- Unidade de controle microprogramada. Memória de microcontrole com 1024 palavras de 88 bits e ciclo de microinstrução de 250 nanossegundos.

- Possibilidade de expansão do microprograma. As versões 1.0 e 1.1 do microprograma ocupam, respectivamente, 50% e 60% da memória de microcontrole. O restante desta memória está disponível para a implementação de instruções mais adequadas para as aplicações específicas do computador, instruções estas a serem definidas posteriormente. Oito dos 88 bits da palavra de microcontrole também não são utilizados nas versões 1.0 e 11 do microprograma.

- Barramentosinternos de 16 bits.

- Unidade de lógica e aritmética que realiza as operações básicas tanto com 8 como com 16 bits em paralelo. Os números inteiros com sinal são representados em complemento de dois.

- Processamento de "bytes" (8 bits), palavras (16 bits) e palavras longas (32 bits).

. Multiplicação e divisão envolvendo números de 16 bits, representados em complemento de dois, através de unidade aritmética opcional (ASTROM).

. Operações aritméticas envolvendo números de ponto flutuante (palavras longas de 32 bits) através de unidade aritmética opcional (ASTROM).

. Doze registros (todos de 16 bits) alteráveis por programação:

RA, RB, RC, RD, RE e RF - registros de uso geral.
SP - ponteiro da pilha de programa.
PC - contador de programa.
SPS - ponteiro da pilha do sistema.
LP - limite das pilhas.
MK - máscara de interrupções.
PSW - palavra de "status" do processador.

. Gerenciamento de duas pilhas residentes na memória, com deteção automática de violação dos seus limites.

. Dezesseis níveis de prioridade de interrupção.

. Código de operação das instruções expansível. Formato das instruções é variável: instruções com 16, 32 e 48 bits. Doze modos de endereçamento. Prefetch de um nível de instrução.

. Protocolo de comunicação no barramento (BASIS) que liga a UCP/ASTROP com memória e controladores de periféricos, do tipo mestre/escravo, totalmente assíncrono e com registros dos controladores de periféricos mapeados na memória.

. Acesso à memória e registros de controladores de periféricos por "byte" (8 bits) ou por palavra (16 bits).

. Endereçamento direto de "bytes" e palavras até 32 kpalavras ou 64 k"bytes" (K=1024).

. Permissão de acesso direto à memória por controladores de periféricos, através de roubos de ciclos da UCP/ASTROP.

2.2.1 - CONJUNTO DE INSTRUÇÕES

A UCP/ASTROP possui um conjunto de instruções com 156 e 170 códigos de operação, respectivamente, nas versões 1.0 e 1.1 do microprograma nela residente e doze modos de endereçamento. Trinta e dois destes códigos de operação estão diretamente relacionados com a utilização da unidade aritmética opcional ASTROM. O código de operação das instruções é do tipo expansível. Existem instruções para manipular "bytes", palavras e palavras longas.

As operações lógicas e aritméticas podem ser realizadas com os dados armazenados em qualquer combinação no caso de dois operandos, nos registros de uso geral, nas posições de memória e nos registros dos controladores de periféricos (que são mapeados na memória). O que torna desnecessária a preocupação, ao nível de programação, em trazer dados da memória ou dos registros dos controladores de periféricos para poder operar com eles e facilita a confecção de programas em linguagem do tipo "assembly". Devido aos modos de endereçamento as instruções apresentam formato variável, podendo ter 16, 32 ou 48 bits.

O conjunto de instruções da UCP/ASTROP (versões 1.0 e 1.1 do microprograma) encontra-se detalhado no Capítulo 4.

2.2.2 - REGISTROS DE USO GERAL

A UCP/ASTROP contém 6 registros de uso geral (Figura 2.3) que podem ser utilizados no armazenamento de dados (com 8, 16 e 32 bits), índices e endereços (ambos com 16 bits). O "reset" da UCP/ASTROP não afeta o conteúdo destes registros. O "byte" mais significativo de um registro da UCP/ASTROP (bits 15-8) é chamado de "byte" H ("high") e o menos significativo (bits 7-0) de "byte" L ("low").

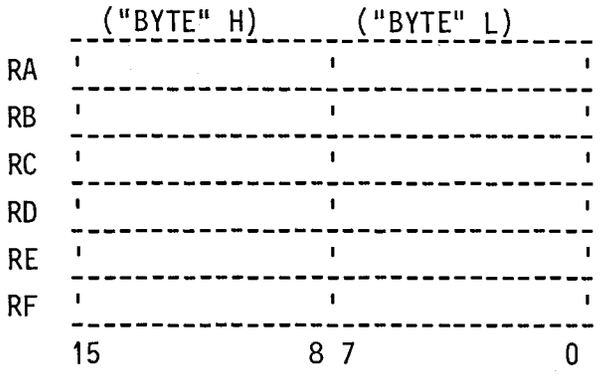
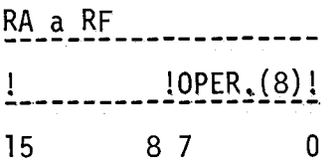


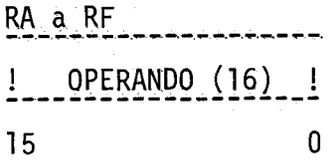
Fig. 2.3 - Registros de uso geral.

A UCP/ASTROP possui instruções que manipulam "bytes" (8 bits), palavras (16 bits) e palavras longas (32 bits). Se o operando (ou um dos operandos) de uma instrução estiver nos registros de uso geral, então:

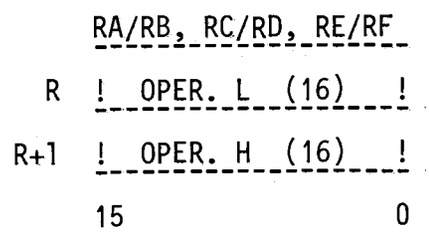
a) Se ele for um "byte", estará nos 8 bits menos significativos do registro.



b) Se ele for uma palavra, será o conteúdo do próprio registro.



c) Se ele for uma palavra longa estará em um par de registros que pode ser os pares RA e RB (referenciando-se RA), RC e RD (referenciando-se RC), RE e RF (referenciando-se RE). Os 16 bits menos significativos (operando "Low" - L) estarão no registro RA, RC ou RE, enquanto os 16 bits mais significativos (operando "high" - H) estarão no registro RB, RD, ou RF.



2.2.3 - REGISTRO CONTADOR DE PROGRAMA

O registro PC (Figura 2.4) contém o endereço da próxima instrução a ser executada. Ele apresenta o bit menos significativo preso em zero, não sendo possível modificar esta condição por programação. O acesso ao seu conteúdo é sempre feito com instruções que manipulam palavras. Após o "reset" da UCP"ASTROP o registro PC recebe o valor armazenado na posição de memória "FF82"H (vetor de "reset").

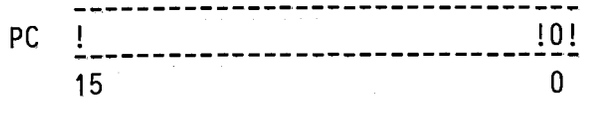


Fig. 2.4 - Registro PC.

2.2.4 - PILHAS

A UCP/ASTROP gerencia, com o auxílio dos registros SP, SPS e LP (Figura 2.5), o conteúdo de duas estruturas de dados do tipo pilha ("Last-in, first-out") residentes na memória do ASTROP: pilha de programa e pilha do sistema. Ambas crescem no sentido dos endereços menos significativos da memória e decrescem no sentido dos endereços mais significativos.

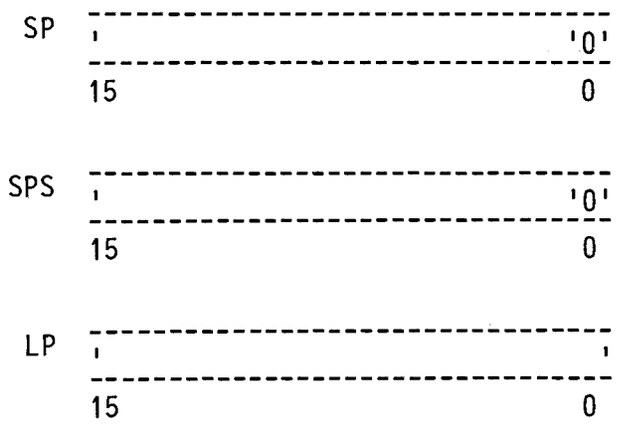


Fig. 2.5 - Registros SP, SPS e LP.

O registro SP aponta para o topo da pilha de programa, ou seja, armazena o endereço do último dado empilhado. O bit menos significativo deste registro está sempre preso em zero, não sendo possível modificar esta condição por programação. O acesso ao seu conteúdo é feito por instruções que manipulam "bytes" ou palavras. Além de armazenar dados, parâmetros, etc.; a pilha de programa é usada pelo procedimento da UCP/ASTROP para desvio e retorno de sub-rotina. Após o "reset" da UCP/ASTROP o registro SP recebe o valor "F700"H.

O registro SPS aponta para o topo da pilha do sistema. O bits menos significativo deste registro também está preso em zero, não sendo possível modificar esta condição por programação. A pilha do sistema é usada apenas para salvar os conteúdos dos registros PC e PSW de um programa, antes de ocorrer a busca de novos conteúdos para os registros PC e PSW referentes ao procedimento da UCP/ASTROP para atendi

mento de um "trap" ou de uma interrupção pendente. O acesso ao conteúdo do registro SPS é sempre feito com instruções que manipulam palavras. Após o "reset" da UCP/ASTROP o registro SPS recebe o valor inicial "F800"H.

O registro LP armazena o endereço limite inferior até onde as duas pilhas residentes na memória do ASTROP podem crescer. Após o "reset" da UCP/ASTROP o registro LP recebe o valor inicial "F600"H. O endereço limite superior (base das pilhas) é prefixado em "F7FF"H. O acesso ao conteúdo do registro LP é sempre feito por instruções que manipulam palavras.

Se, ao acessar a pilha de programa, para leitura ou escrita, o endereço utilizado pela UCP/ASTROP for maior do que "F7FF"H ou menor do que o conteúdo do registro LP, um "trap" de violação dos limites da pilha vai ocorrer (ver Seção 2.2.7). Por outro lado, se o endereço utilizado para acessar a pilha do sistema estiver fora desses limites permitidos, a UCP/ASTROP automaticamente entrará no estado de "HALT". Note-se que estas ações ocorrerão apenas durante o acesso às pilhas e que não são causadas diretamente pelos conteúdos dos registros SP e SPS, mas pelos endereços deles resultantes. Por exemplo, o registro SPS pode armazenar o valor "F800"H (maior do que a base "F7FF"H), pois isto não causa um "HALT" na UCP/ASTROP. Se, após isto, a pilha do sistema for utilizada para uma operação de empilhamento ("push"), o endereço decorrente será "F7FE"H, o que está dentro dos limites permitidos. Entretanto, se esta pilha for utilizada para uma operação de desempilhamento ("pop"), o endereço empregado para acessá-la será o próprio "F800"H, o que força a UCP/ASTROP a entrar no estado de "HALT".

2.2.5 - PALAVRA DE "STATUS" DO PROCESSADOR

A palavra de "status" do processador (PSW) contém as informações sobre o "status" corrente da UCP/ASTROP, o que compreende (Figura 2.6):

1) Base do programa (B3 a B0)

É um valor binário que é adicionado aos quatro bits mais significativos de qualquer endereço gerado pela UCP/ASTROP, exceto:

a) nos acessos às pilhas residentes na memória do ASTROP;

b) nos acessos aos últimos 2k "bytes" da memória (endereços "F800"H a "FFFF"H) que são reservados para os registros dos controladores de periféricos e os vetores de "reset", de "traps" e de interrupções;

c) durante a execução de algumas instruções especiais.

Esta base torna disponível um mecanismo eficiente para realocação, na memória do ASTROP, de programas (ou parte de programas) na forma de código objeto. Como exemplo suponha-se um programa em código objeto, montado para ser executado quando armazenado na memória do ASTROP a partir da posição "2000"H. Este mesmo código objeto poderá ser armazenado, por exemplo, a partir da posição "7000"H e ser executado com valor de base igual a "5"H. A qualquer endereço de instrução ou dado gerado na execução deste programa, a UCP/ASTROP adicionará automaticamente o valor da base de programa, tornando desnecessária a realocação destes endereços no código objeto. O acesso correto a parâmetros passados pela pilha de programa bem como aos conteúdos dos registros dos controladores de periféricos está garantido, já que a estes endereços não será adicionado o valor da base.

2) Nível mínimo de interrupção (S3 a S0)

Sendo i = 8xS3 + 4xS2 + 2xS1 + S0, qualquer pedido de interrupção não-mascarado PIi (Seção 2.2.6), com i<j, será ignorado pela UCP/STROP, até que o nível mínimo de interrupção seja modificado de forma que i passe a ser maior ou igual a j e desde que não haja um outro pedido de interrupção pendente de nível maior do que i.

3) Bit de "trap" de"breakpoint" (B)

Este bit pode ser modificado apenas sob o controle do programa e é usado na depuração do "software", pois implementa uma maneira eficiente (ao nível de "hardware") de realizar "breakpoints" e executar programas passo a passo. Quando este bit é feito igual a 1 (um), um "trap" de "breakpoint" ocorrerá.

4) Sete códigos de condição (T1, T2, T3, C, V, Z e S)

Três destes códigos de condição (T1, T2 e T3) podem ser usados livremente pelos programas para sinalizar regiões críticas, indicar resultados de sub-rotinas, etc., já que são afetados apenas pelas instruções que comandam explicitamente estas modificações.

Os outros quatro códigos de condições (C, V, Z e S) são modificados automaticamente pela UCP/ASTROP, de acordo com o resultado da última operação lógica ou aritmética executada. Eles encerram os seguintes significados:

C = "1", se a operação causa a geração de um bit de transporte;
"0", caso contrário.

V = "1", se a operação resulta em um "overflow" aritmético;
"0", caso contrário.

Z = "1", se o resultado for zero;
"0", caso contrário.

S = "1", se o resultado for negativo;
"0", caso contrário.

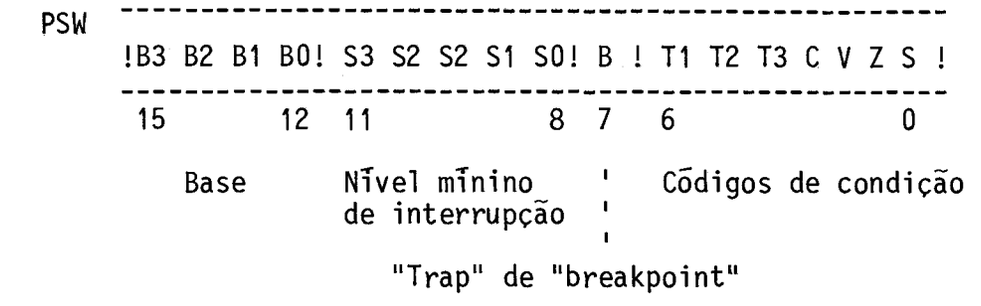
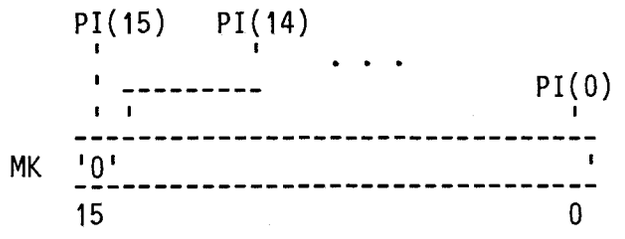


Fig. 2.6 - Registro PSW.

Após o "reset" da UCP/ASTROP, o registro PSW recebe o valor armazenado na posição "FF80"H da memória do ASTROP.

2.2.6 - INTERRUPÇÕES

O registro máscara de interrupções (Figura 2.7), como o próprio nome indica, mascara os pedidos de interrupção originados nas linhas PI (15-00) da UCP/ASTROP. O bit 15 deste registro está preso em 0 (zero), não sendo possível mascarar a linha de mais alto nível PI (15). Após o "reset" da UCP ASTROP o registro MK recebe o valor "0000"H, o que habilita o reconhecimento de todos os seus dezesseis níveis de interrupções. O acesso ao registro MK é sempre feito por instruções que manipulam palavras.



Obs.: MKi = "0" --> Reconhece interrupção da linha PIi.
"1" --> Máscara interrupção da linha PIi.

Fig. 2.7 - Registro MK.

Um pedido de interrupção de uma linha PIi só será reconhecido pela UCP/ASTROP, apenas se o bit i da máscara de interrupções armazenada no registro MK for "0". Caso contrário o pedido é ignorado. Note-se que, além de não estar mascarado, um pedido de interrupção PIi só será atendido pela UCP/ASTROP se:

a) não existir um outro pedido de interrupção pendente, não mascarado e de nível maior do que i,

b) o nível mínimo de interrupção codificado na palavra de "status" do processados (PSW) for menor ou igual a i.

No processo de atendimento de interrupções pela UCP/ASTROP estão envolvidos os vetores de interrupção residentes na memória do ASTROP (Figura 2.8). Neste processo, inicialmente a UCP/ASTROP salva na pilha do sistema os conteúdos dos registros PC e PSW (nesta ordem). A seguir, busca na memória em endereços prefixados para cada nível de interrupção, os novos conteúdos para os registros PC e PSW.

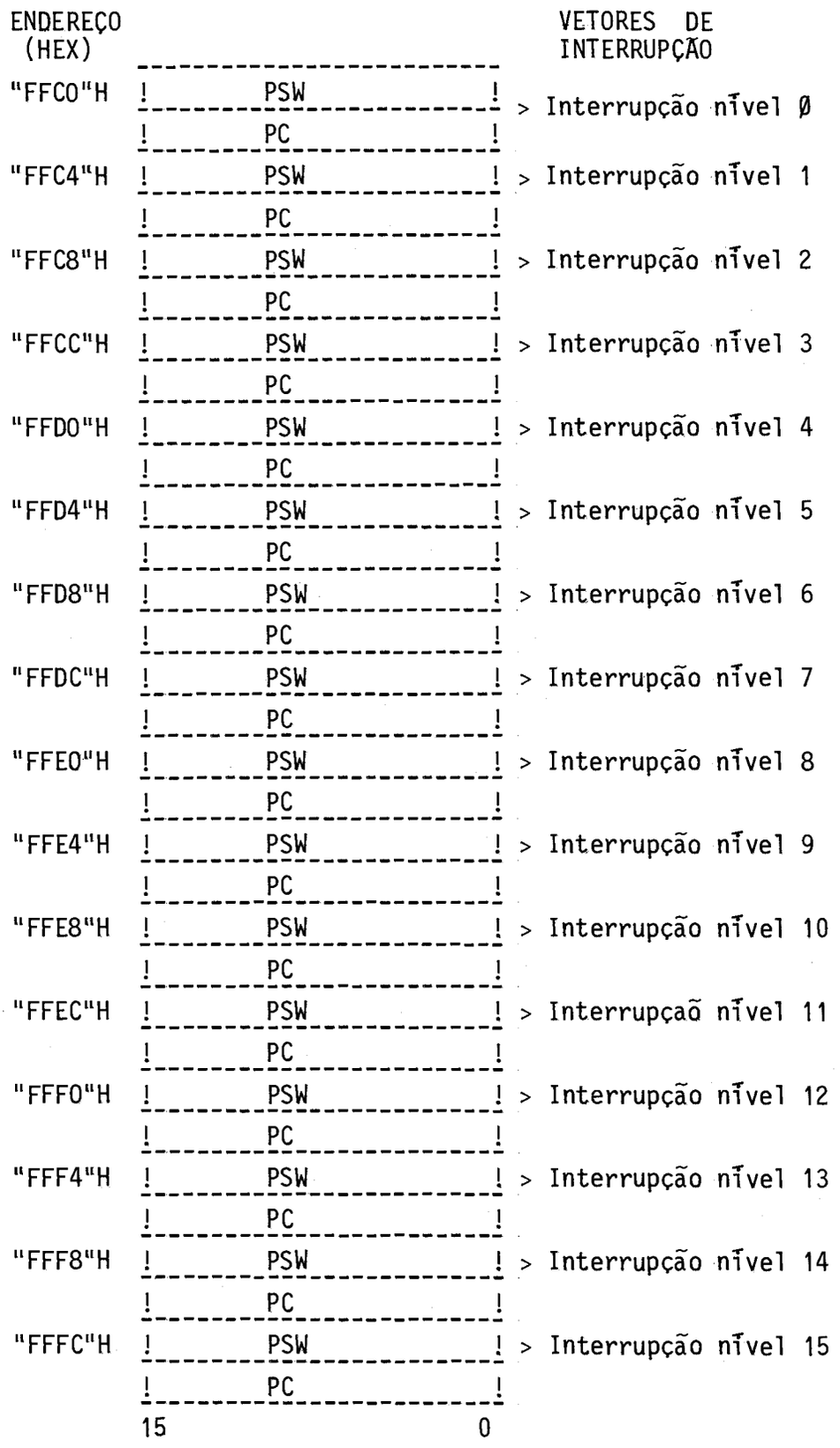


Fig. 2.8 - Vetores de interrupção

2.2.7 - "TRAPS"

A UCP/ASTROP possui sete "traps" que são automaticamente detetados por "hardware":

1) "Trap" de violação dos limites da pilha

Este "trap" ocorrerá se o endereço utilizado pela UCP/ASTROP para acessar a pilha de programa for maior do que "F7FF"H ou menor do que o conteúdo do registro LP.

2) "Trap" de endereçamento ímpar de palavra

O acesso à memória do ASTROP pela UCP/ASTROP pode se dar tanto por "bytes" (8 bits) como por palavras (16 bits). Os endereços para ler e escrever palavras na memória necessariamente têm de ser valores pares (bit menos significativo igual a zero). O mesmo deve ocorrer no acesso aos registros dos controladores de periféricos. Se o fluxo de um programa levar a UCP/ASTROP à situação de ter de acessar uma palavra com endereço ímpar (bit menos significativo igual a 1), então um "trap" de endereçamento ímpar de palavra irá ocorrer.

3) "Trap" de "timeout" no BASIS

Tanto em uma operação de leitura como em uma operação de escrita a UCP/ASTROP, atuando como mestre do BASIS, aguarda no máximo 1 milissegundo pela resposta do dispositivo escravo envolvido na operação. Caso o escravo não responda neste tempo hábil, a UCP/ASTROP evita um possível"deadlock" abortando a ciclo de transferência de dados e atendendo automaticamente a um "trap" de "timeout" do BASIS. Note-se que a ocorrência de "timeout" no BASIS geralmente se dá devido ao endereçamento de uma posição de memória ou de registro de controlador de periférico inexistente (erro de Programação) e não por defeito de circuito.

4) "Trap" de erro de paridade no BASIS

Este "trap" ocorrerá se houver um erro de paridade no dado acessado durante uma operação de leitura, pela UCP/ASTROP, da memória ou de um registro de controlador de periférico.

5) "Trap" da unidade aritmética ASTROM

A unidade aritmética ASTROM deteta, durante a realização de suas operações aritméticas, as seguintes ocorrências anormais:

. erro de conversão,

. "underflow",

. "overflow",

. divisão por zero.

Estas quatro ocorrências, acrescidas de uma quinta:

. unidade aritmética desligada,

causam o atendimento pela UCP/ASTROP de um "trap" da unidade aritmética ASTROM, durante a execução das instruções que se utilizam do ASTROM.

6) "Trap" de "breakpoint"

A ocorrência deste "trap" se dá sob controle do programa quea UCP/ASTROP executa, pois ele é causado pelo estado do bit 8 da palavra de "status" do processador. Ver Seção 2.2.5.

7) "Trap" de código de operação inválido

a UCP/ASTROP tente executar um código de operação de uma instrução ou modo de endereçamento não-definido ou não-permitido.

O processo de atendimento destes sete "traps" pela UCP/ASTROP é idêntico ao descrito para atendimento de interrupções (Seção 2.2.6). Os vetores de "traps" (análogos aos vetores de interrupções) necessitam estar armazenados nas posições de memória definidas na Figura 2.9. Note-se nesta figura que, além do vetor de "reset" abordado na Seção 2.2.8, existem posições de memória prefixadas também para oito vetores de "traps" relacionados com a execução da instrução TRAP.

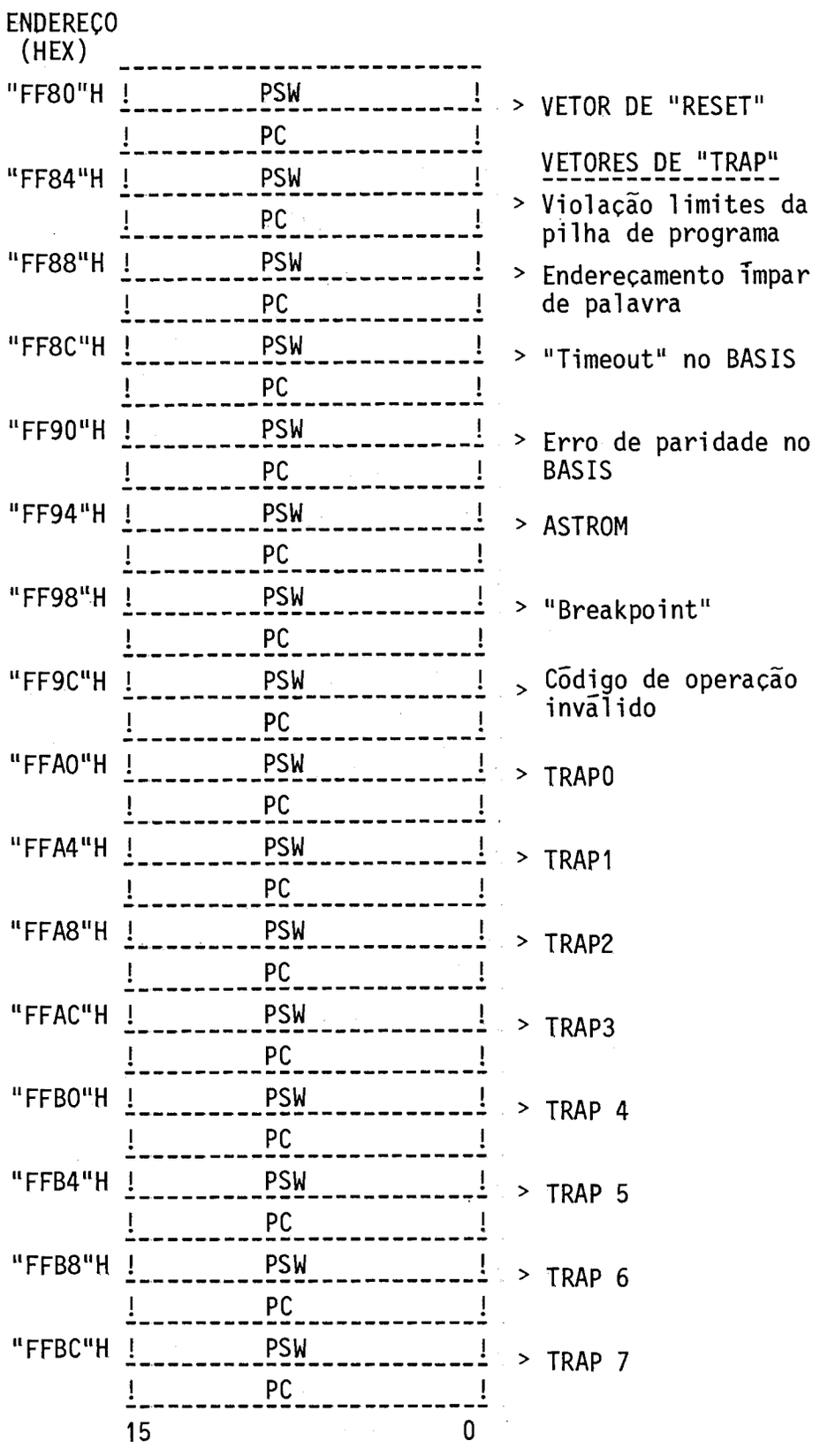


Fig. 2.9 - Vetores de "reset" e de "traps".

2.2.8 - "RESET"

O "reset" do computador ASTROP, incluindo a UCP/ASTROP, pode ser causado de três maneiras:

a) sempre que os circuitos do computador são energizados ("power-up"),

b) pela execução da instrução RESET,

c) através do acionamento do "pushbutton" existente no painel do computador para esta finalidade.

Qualquer que seja a sua origem, a duração do "reset" (sinal $\overline{INI}$ do BASIS ativado) é de 1 segundo. Após o "reset" o contexto interno da UCP/ASTROP passa a ser o da Tabela 2.2 e ela entra no estado de "HALT" (Seção 2.2.9).

TABELA 2.2

CONTEXTO DA UCP/ASTROP APÓS O "RESET"

REGISTRO	CONTEÚDO	OBSERVAÇÃO
RA, RB, RC, RD, RE e RF	---	Não são afetados
SPS	"F800"H	Base das pilhas prefixada em "F7FF"H
SP	"F700"H	
LP	"F600"H	
PC	("FF82"H)	Vetor de "reset" (Figura 2.9)
PSW	("FF80"H)	Vetor de "reset" (Figura 2.9)
MK	"0000"H	Habilita reconhecimento de todas as interrupções

Observação: ("AAAA"H) significa conteúdo da posição de memória AAAA.

2.2.9 - ESTADOS DA UCP/ASTROP

A UCP/ASTROP pode estar em três estados básicos (Figura 2.10):

1) "RUN" - executando instruções.

2) "HALT" - parada.

3) "WAIT" - aguardando uma interrupção.

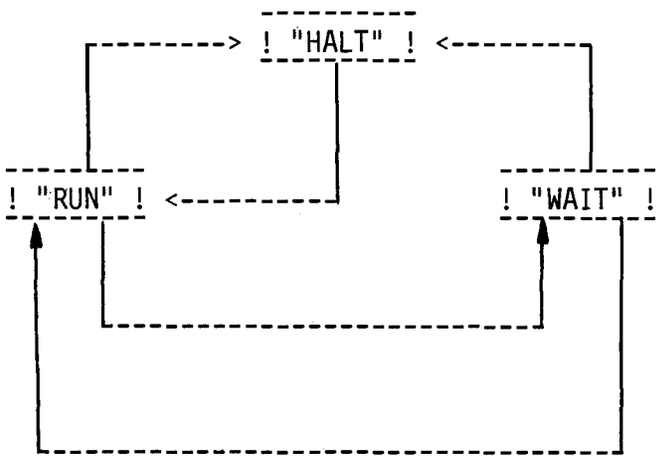


Fig. 2.10 - Estados básicos da UCP/ASTROP.

As causas de mudanças de estado da UCP/ASTROP são:

a) "RUN" para "HALT":

- execução da instrução HALT;

- "reset" da UCP"ASTROP;

- ocorrência dos seguintes "traps" durante o acesso da pilha do sistema:

. violação dos limites da pilha,

. endereçamento ímpar de palavra,

. "timeout" no BASIS,

. erro de paridade no BASIS;

- acionamento do "push-button" PARE existente no painel do computador ASTROP.

b) "HALT" para "RUN":

- acionamento do "push-button" EXECUTE existente no painel do computador ASTROP para esta finalidade.

c) "RUN" para "WAIT":

- execução da instrução WAIT.

d) "WAIT" para "RUN":

- atendimento de uma interrupção.

e) "WAIT" para "HALT":

- acionamento do "push-button" PARE existente no painel do computador ASTROP.

2.2.10 - PRIORIDADES DE ATENDIMENTO DE OCORRÊNCIAS

Seguem, listadas em ordem decrescente de prioridade de atendimento pela UCP/ASTROP, as ocorrências de "reset", "traps", interrupções e mudança do estado da UCP/ASTROP para "HALT":

Maior prioridade → "Reset"
"Trap" de violação dos limites da pilha
"Trap" de endereçamento ímpar de palavra
"Trap" de "timeout" no BASIS
"Trap" de erro de paridade no BASIS
"Trap" da unidade aritmética ASTROM
Mudança para o estado "HALT"
"Trap" de "breakpoint"
Interrupção de nível 15
.
.
.
Interrupção de nível 0
"Trap" de código de operação inválido

Menor prioridade → "Trap" das instruções TRAPn

2.3 - MEMÓRIA

A memória do ASTROP, como exposto na Seção 2.1, é a unidade funcional com menor controle sobre o BASIS: age sempre como escravo em qualquer transação e não tem capacidade de interromper o processamento da UCP/ASTROP.

2.3.1 - ORGANIZAÇÃO DA MEMÓRIA

A memória pode ser vista como uma série de posições com um número (endereço) alocado a cada posição que corresponde a um "byte" de 8 bits. O número máximo de "bytes" endereçáveis (16 linhas de endereço no BASIS) é de 65536 "bytes".

Por outro lado, a memória é capaz também de ser acessada, para leitura ou escrita, em duas posições contíguas ("bytes") ao mesmo tempo, possibilitando assim o armazenamento de palavras de 16 bits. O número de palavras endereçáveis é, portanto, igual à metade do

de "bytes": 32768. Analogamente aos registros da UCP/ASTROP, os dois "bytes" que compõem uma palavra da memória chamam-se "byte" H ("high") e "byte" L ("low"), como mostrado na Figura 2.11. Note-se nesta figura que os "bytes" H correspondem a endereços ímpares e os "bytes" L a endereços pares. Palavras devem sempre ser acessadas com endereços pares.

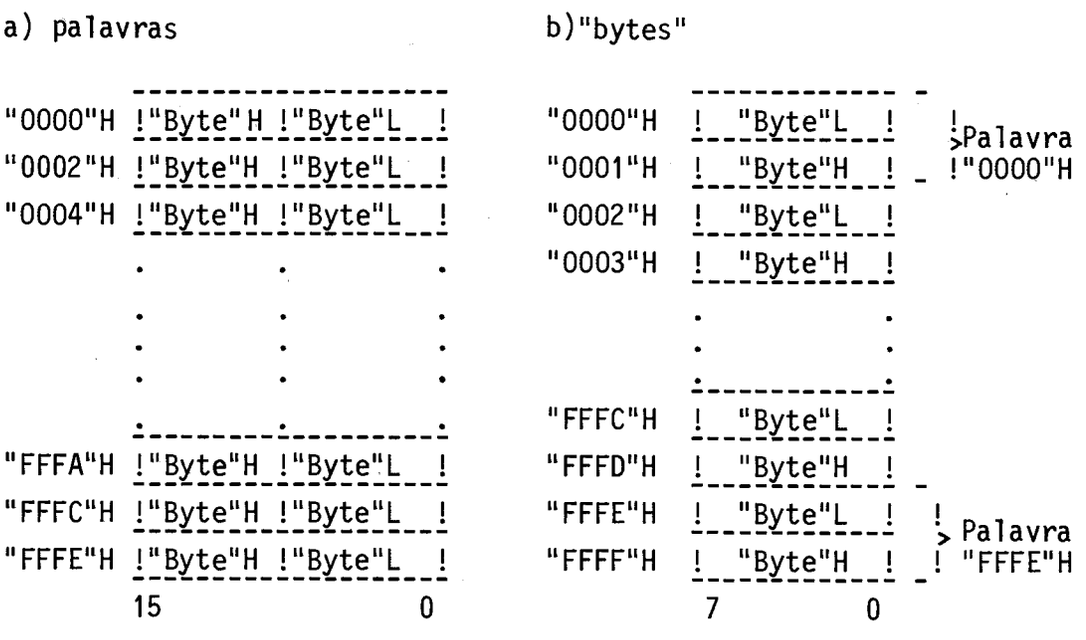


Fig. 2.11 - Endereçamento de palavras e "bytes".

2.3.2 - ALOCAÇÃO DA MEMÓRIA

A capacidade de endereçamento de memória no BASIS encontra-se dividida em duas partes (Figura 2.12). A primeira delas (endereços "0000"H a "F7FF"H - 62 kbytes") composta de posições de memória propriamente ditas onde devem residir os programas em execução e as pilhas de programa e do sistema. A outra (endereços "F800"H a "FFFF"H-

CAPÍTULO 3

MODOS DE ENDEREÇAMENTO

Os dados armazenados na memória e nos registros da UCP/ASTROP devem ser acessados e manipulados. Isto é feito pelas instruções do computador ASTROP que, de forma geral, indicam:

a) a função a ser realizada (código de operação),

b) o(s) registro(s) a ser(em) empregado(s) no processo de localização do(s) operando(s) fonte e/ou destino,

c) o(s) modo(s) de endereçamento que especifica(m) como o(s) registro(s) deve(m) ser utilizado(s).

Apenas os registros, RA, RB, RC, RD, RE, RF, SP e PC entram na definição dos modos de endereçamento das instruções do computador ASTROP.

Os registros de uso geral (RA a RF) podem ser utilizados com uma instrução como:

a) acumuladores: o operando está no próprio registro;

b) ponteiros: o conteúdo do registro é o endereço efetivo do operando que está na memória;

c) índices: o conteúdo do registro e a palavra que segue à instrução são somados para determinar o endereço efetivo do operando que está na memória, ou para determinar o endereço na memória onde se encontra o endereço efetivo do operando que também está na memória.

O registro SP é o ponteiro da pilha do programa que, além de poder armazenar dados, parâmetros de sub-rotinas, etc., é utilizada automaticamente pela UCP/ASTROP no seu procedimento de desvio e retorno de sub-rotinas.

O registro PC é usado automaticamente pela UCP/ASTROP como contador de programa. Podendo também ser empregado como ponteiro e índice na determinação de endereços efetivos de operandos localizados na memória.

Os dados acessados e manipulados no computador ASTROP podem ser "bytes" de 8 bits (referenciados como OPER. (8)), palavras de 16 bits (referenciadas como OPERANDO (16)), ou palavras longas de 32 bits (referenciadas como OPER.L(16) a parte menos significativa e OPER. H (16), a parte mais significativa).

Os modos de endereçamento das instruções do ASTROP são especificados pelo registro associado (RA a RF, SP ou PC) e pelo modo de endereçamento propriamente dito (0, 1, 2 ou 3). Na Figura 3.1 é mostrado, como exemplo, o formato da instrução com um operando, onde os bits 15 a 5 determinam o código de operação da instrução, os bits 4 a 3 o modo (Tabela 3.1) e os bits 2 a 0 o registro associado (Tabela 3.2). Na Tabela 3.3 encontram-se listados os modos de endereçamento que são abordados no restante deste capítulo.

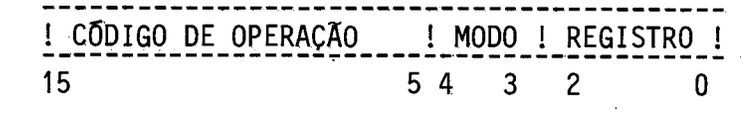


Fig. 3.1 - Formato das instruções com um operando.

TABELA 3.3

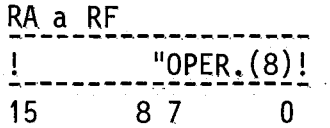
REGISTRO ASSOCIADO	MODO	MODO DE ENDEREÇAMENTO
RA	0	Registro direto
.	1	Registro indireto
.	2	Registro indexado
RF	3	Registro indexado indireto
SP	0	Ponteiro da pilha
	1	Ponteiro autodecrementado
	2	Ponteiro autoincrementado
	3	Topo da pilha
PC	0	Imediato
	1	Absoluto
	2	Relativo
	3	Relativo indireto

3.1 - MODOS DE ENDEREÇAMENTO QUE REFERENCIAM OS REGISTROS DE USO GERAL

3.1.1 - REGISTRO DIRETO

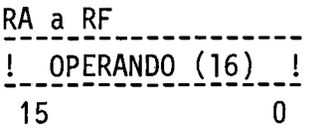
Neste modo de endereçamento o operando está no próprio registro de uso geral, como segue:

a) Acesso a "byte":

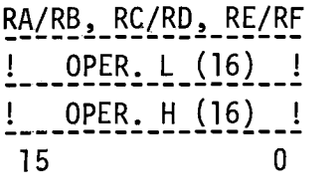


Observação: o "byte" mais significativo não é afetado.

b) Acesso à palavra:



c) Acesso à palavra longa:



Observação: Os pares de registros permitidos são RA e RB (declarando RA), RC e RD (declarando RC), RE e RF (declarando RE). Se forem declarados os registros RB, RC ou RF um "trap" de código de operação inválido irá ocorrer.

3.1.2 - REGISTRO INDIRETO

Neste modo de endereçamento o registro de uso geral contém o endereço efetivo do operando que está na memória (Figura 3.2). O acesso à palavra longa no modo de endereçamento registro indireto resulta na ocorrência de um "trap" de código de operação inválido.

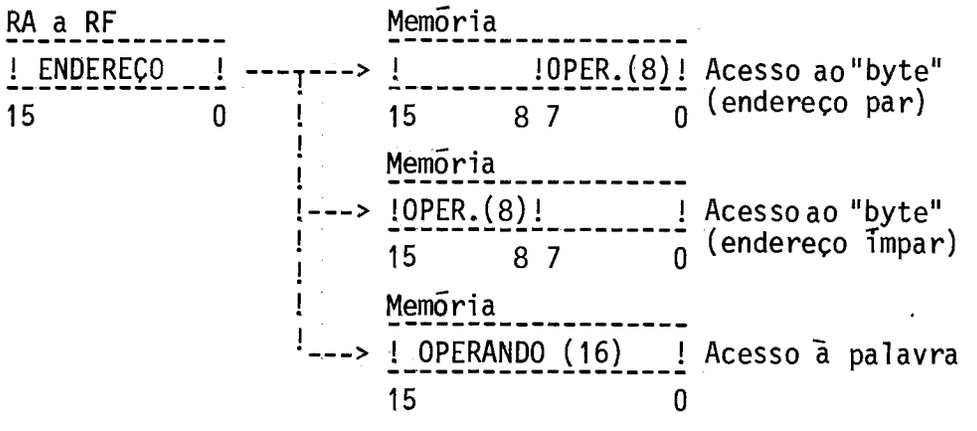


Fig. 3.2 - Modo de endereçamento: registro indireto.

3.1.3 - REGISTRO INDEXADO

Neste modo de endereçamento o conteúdo de um dos registros de uso geral, somado à palavra seguinte à palavra da instrução, fornece o endereço efetivo do operando que está na memória (Figura 3.3). O acesso à palavra longa no modo de endereçamento registro indexado resulta na ocorrência de um "trap" de código de operação inválido. O conteúdo do registro PC é incrementado de dois.

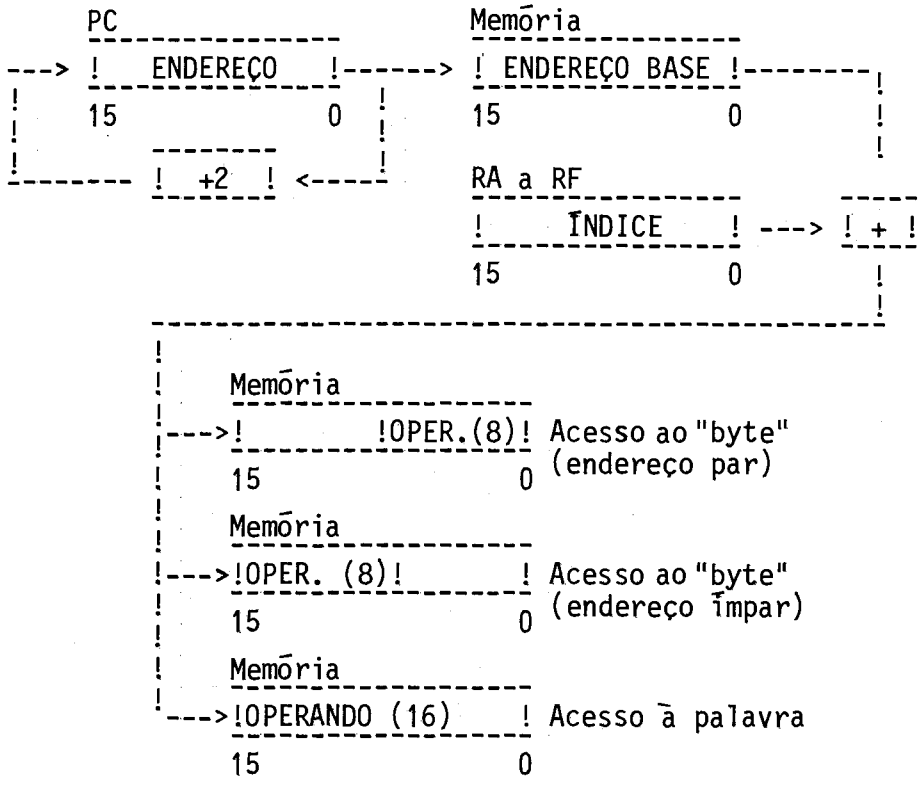


Fig. 3.3 - Modo de endereçamento: registro indexado.

3.1.4 - REGISTRO INDEXADO INDIRETO

Neste modo de endereçamento o conteúdo de um dos registros de uso geral, somado à palavra seguinte à palavra da instrução, fornece o endereço de uma posição de memória onde está o endereço efetivo do operando (Figura 3.4). O acesso à palavra longa no modo de endereçamento registro indexado indireto resulta na ocorrência de um "trap" de código de operação inválido. O conteúdo do registro PC é incrementado de dois.

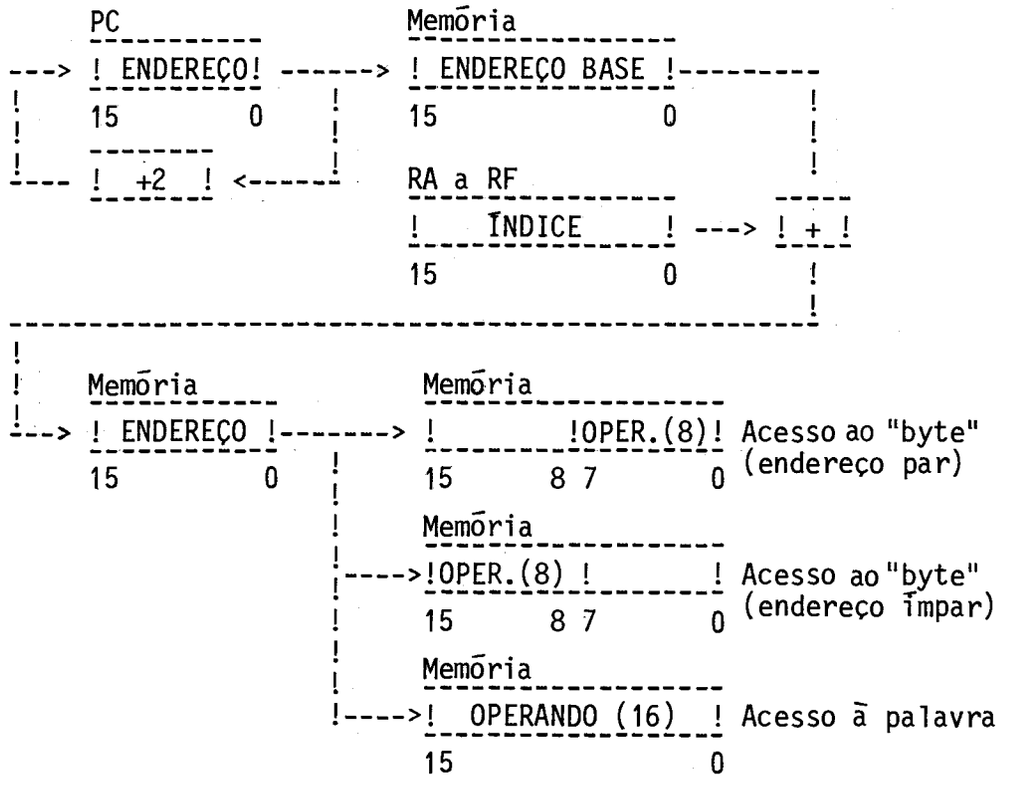


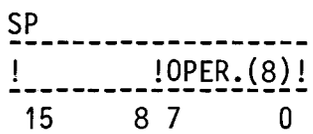
Fig. 3.4 - Modo de endereçamento: registro indexado indireto.

3.2 - MODOS DE ENDEREÇAMENTO QUE REFERENCIAM O REGISTRO PONTEIRO DA PILHA

3.2.1 - PONTEIRO DA PILHA

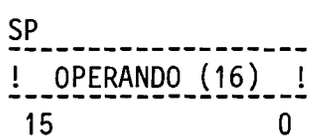
Neste modo de endereçamento o conteúdo do registro ponteiro da pilha é o operando, como se segue:

a) Acesso ao "byte":



Observação: O "byte" mais significativo não é afetado.

b) Acesso à palavra:

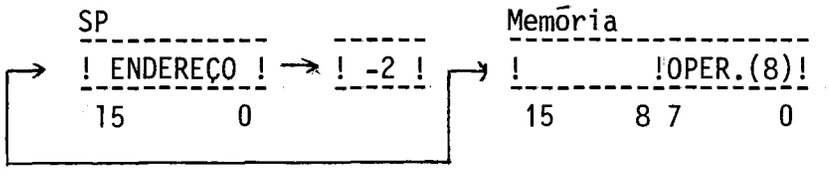


O acesso à palavra longa no modo de endereçamento ponteiro da pilha resulta na ocorrência de um "trap" de código de operação inválido.

3.2.2 - PONTEIRO AUTODECREMENTADO

Neste modo de endereçamento o conteúdo do registro ponteiro da pilha decrementado de dois é o endereço efetivo do operando (Figura 3.5). O conteúdo do registro SP passa a ser o seu conteúdo original decrementado de dois. O acesso à palavra longa no modo de endereçamento ponteiro autodecrementado resulta na ocorrência de um "trap" de código de operação inválido. Este modo de endereçamento normalmente é utilizado para designar um operando destino, o que resulta no empilhamento ("push") de um dado na pilha de programa.

a) Acesso ao "byte":



Observação: O conteúdo do registro SP é sempre par.

b) Acesso à palavra:

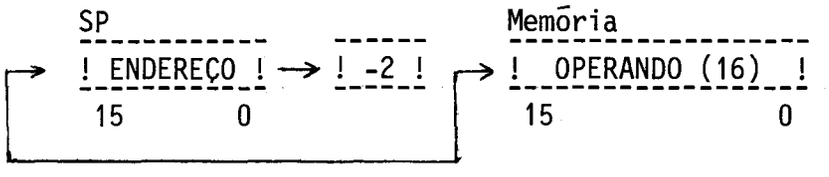
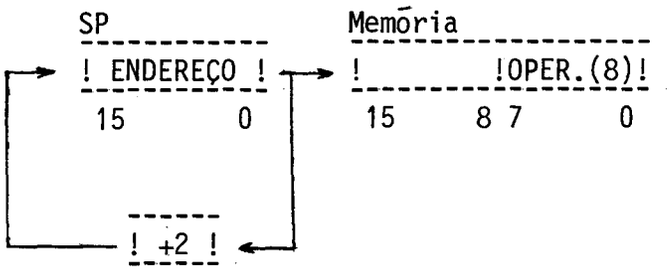


Fig. 3.5 - Modo de endereçamento: ponteiro autodecrementado.

3.2.3 - PONTEIRO AUTOINCREMENTADO

Neste modo de endereçamento o conteúdo do registro ponteiro da pilha é o endereço efetivo do operando (Figura 3.6). O conteúdo do registro SP passa a ser o seu conteúdo original incrementado de dois. O acesso à palavra longa no modo de endereçamento ponteiro autoincrementado resulta na ocorrência de um "trap" de código de operação inválido. Este modo de endereçamento normalmente é utilizado para designar um operando fonte, o que resulta no desempilhamento ("pop") de um dado da pilha de programa.

a) Acesso ao "byte":



Observação: O conteúdo do registro SP é sempre par.

b) Acesso à palavra:

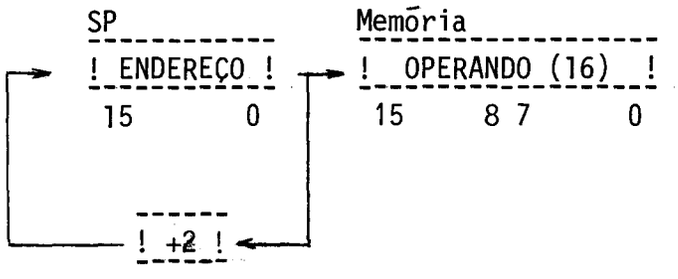
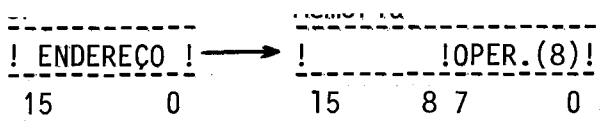


Fig. 3.6 - Modo de endereçamento: ponteiro autoincrementado.

3.2.4 - TOPO DA PILHA

Neste modo de endereçamento o conteúdo do registro ponteiro da pilha é o endereço efetivo do operando (Figura 3.7). O acesso à palavra longa no modo de endereçamento topo da pilha resulta na ocorrência de um "trap" de código de operação inválido.

a) Acesso ao "byte":



Observação: O conteúdo do registro SP é sempre par.

b) Acesso à palavra:

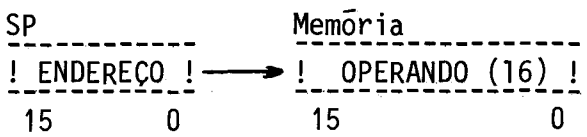


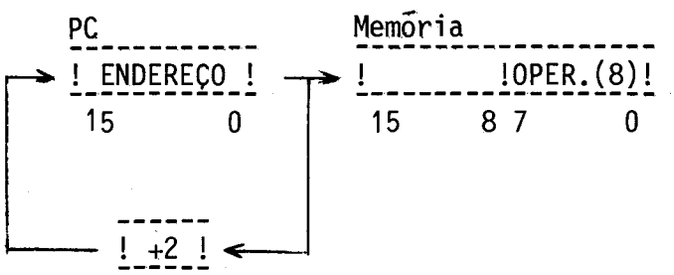
Fig. 3.7 - Modo de endereçamento: topo da pilha.

3.3 - MODOS DE ENDEREÇAMENTO QUE REFERENCIAM O REGISTRO CONTADOR DE PROGRAMA

3.3.1 - IMEDIATO

Neste modo de endereçamento o operando está na palavra seguinte à palavra da instrução (Figura 3.8). O conteúdo do registro PC é incrementado de dois. O acesso à palavra longa no modo de endereçamento imediato resulta na ocorrência de um "trap" de código de operação inválido.

a) Acesso ao "byte":



Observação: O conteúdo do registro PC é sempre par.

b) Acesso à palavra:

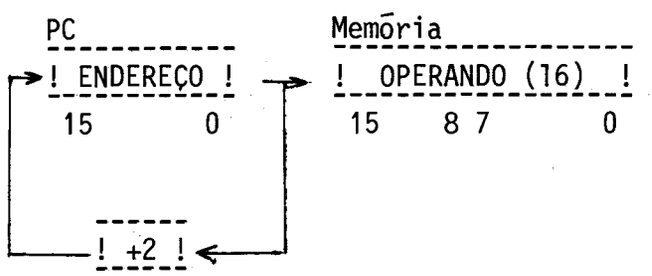


Fig. 3.8 - Modo de endereçamento: imediato.

3.3.2 - ABSOLUTO

Neste modo de endereçamento o conteúdo da palavra seguinte à palavra da instrução é o endereço efetivo do operando (Figura 3.9). O conteúdo do registro PC é incrementado de dois. O acesso à palavra longa no modo de endereçamento absoluto resulta na ocorrência de um "trap" de código de operação inválido.

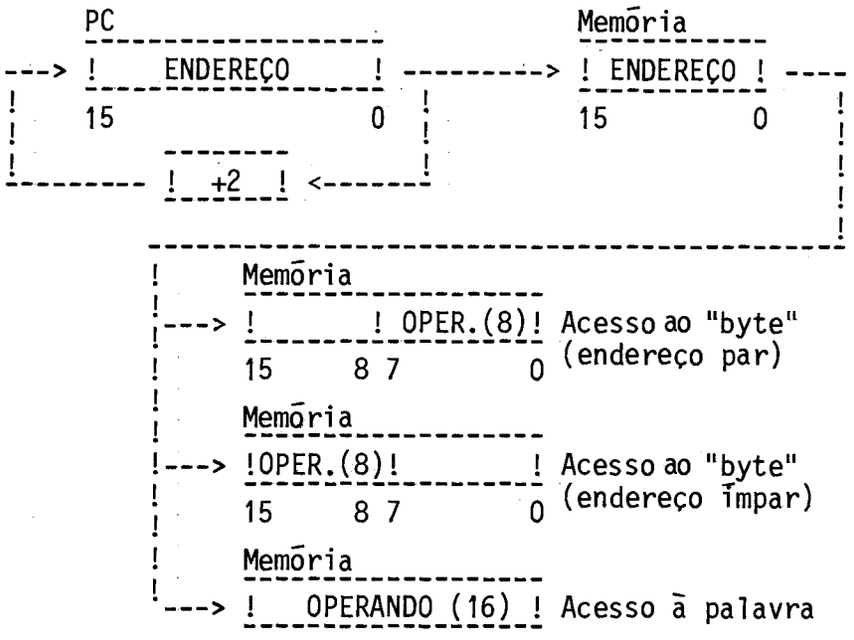


Fig. 3.9 - Modo de endereçamento: absoluto.

3.3.3 - RELATIVO

Neste modo de endereçamento a palavra seguinte à palavra da instrução, somada ao conteúdo do registro contador de programa (já incrementado após o "fetch"), é o endereço efetivo do operando (Figura 3.10). O conteúdo do registro PC é incrementado em dois. O acesso à palavra longa no modo de endereçamento relativo resulta na ocorrência de um "trap" de código de operação inválido.

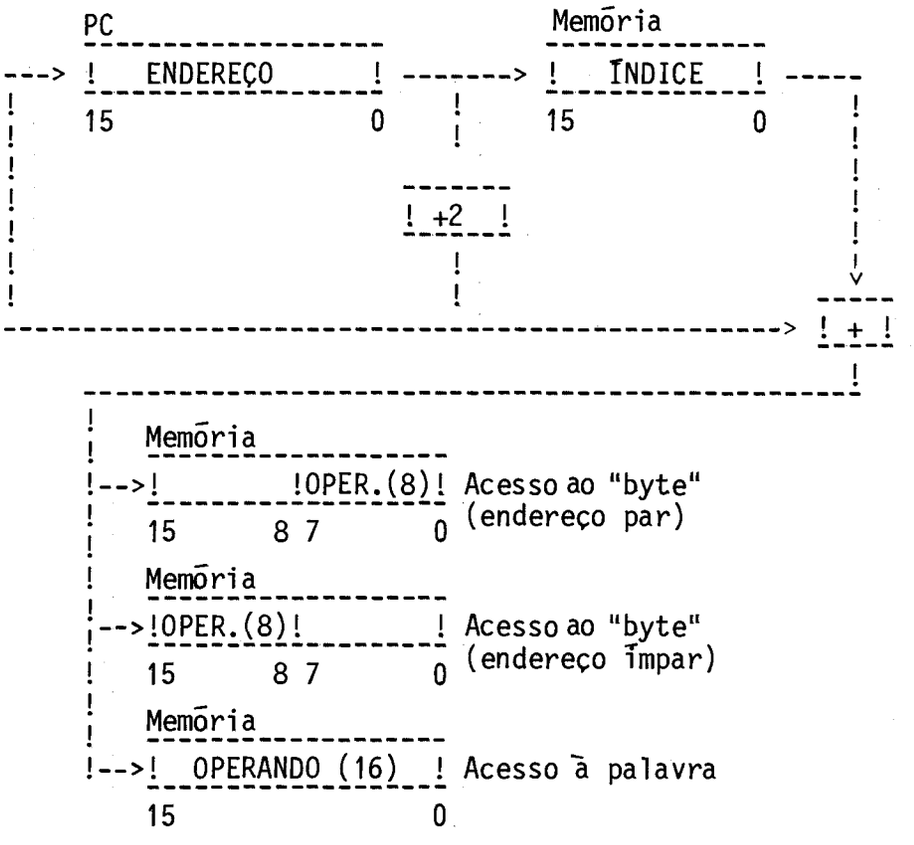


Fig. 3.10 - Modo de endereçamento: relativo.

3.3.4 - RELATIVO INDIRETO

Neste modo de endereçamento a palavra seguinte à palavra da instrução, somada ao conteúdo do registro contador de programa (já incrementado após o "fetch"), é o endereço da posição de memória que contém o endereço efetivo do operando (Figura 3.11). O conteúdo do registro PC é incrementado de dois. O acesso à palavra longa no modo de endereçamento relativo indireto resulta na ocorrência de um "trap" de código de operação inválido.

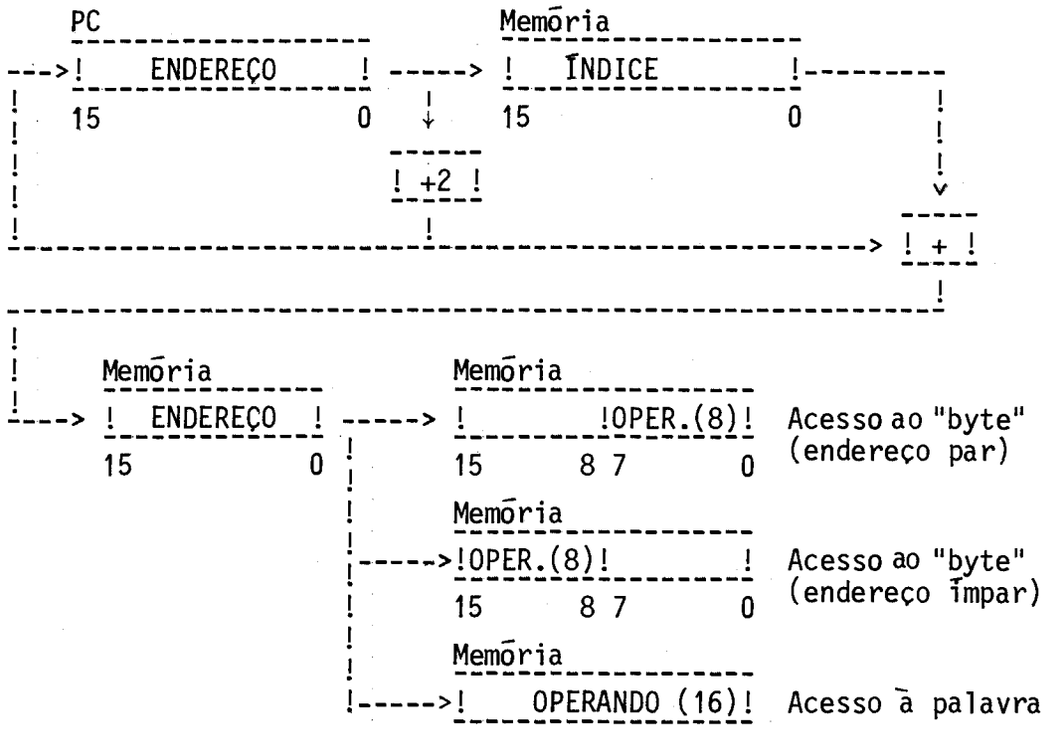


Fig. 3.11 - Modo de endereçamento: relativo indireto.

CAPÍTULO 4

CONJUNTO DE INSTRUÇÕES

Neste capítulo é apresentado, após o formato das instruções, o conjunto de instruções da UCP/ASTROP, o que inclui para cada instrução:

a) o seu mnemônico e código de operação em representação binária,

b) um diagrama que mostra o formato da instrução,

c) uma descrição simbólica de sua execução,

d) seus efeitos sobre os códigos de condição da UCP/ASTROP,

e) uma descrição com comentários.

Dentro do possível, cada instrução é apresentada em uma única página.

Os mnemônicos das instruções estão indicados no canto superior direito de cada página. Quando a instrução que manipula operandos de 16 bits (palavra) possui uma equivalente que manipula operandos de 8 bits ("byte"), ambos os mnemônicos são mostrados. É importante ressaltar que as instruções que manipulam "bytes", necessariamente têm o bit mais significativo (bit 15) igual a 0 (zero) e os seus mnemônicos correspondem aos respectivos menemônicos das instruções que manipulam palavras, acrescidos da letra B. Por exemplo: soma palavras - ADD, soma "bytes" - ADDB.

Na descrição simbólica da execução das instruções os seguintes símbolos são utilizados:

( ) = conteúdo de

<--> = permuta

<-- = torna-se

[-] = negação lógica

[+] = "OU" lógico

[.] = "E" lógico

[*] = "OU-EXCLUSIVO" lógico

fnt = endereço efetivo do operando fonte

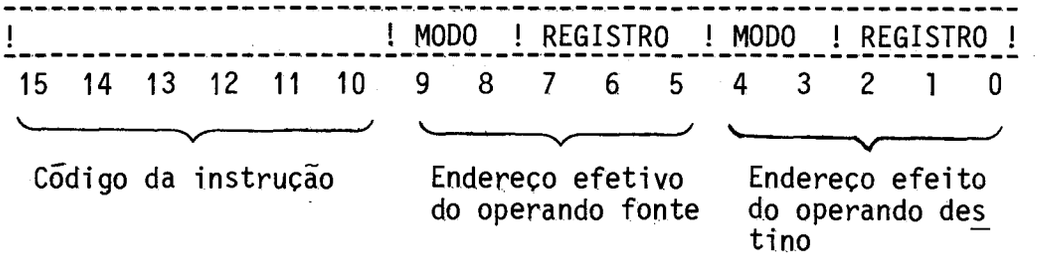
dst = endereço efetivo do operando destino

1/0 = 1 - palavra; 0 - "byte"

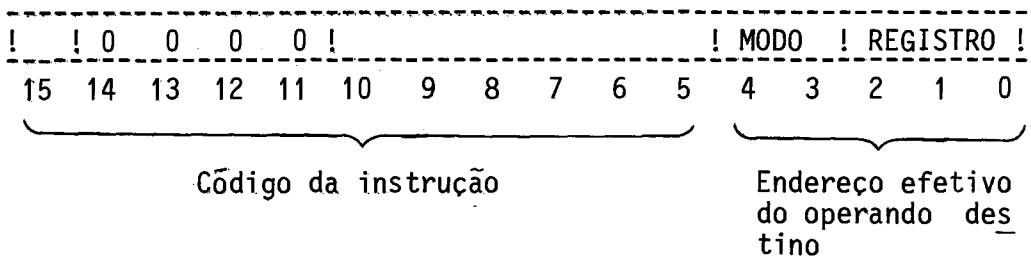
4.1 - FORMATOS DAS INSTRUÇÕES

Os principais formatos das instruções da UCP/ASTROP são apresentados a seguir. Os códigos dos modos e dos registros associados na especificação do modo de endereçamento dos operandos fonte e destino estão, respectivamente, nas Tabelas 3.1 e 3.2. No caso das instruções com dois operandos, a UCP/ASTROP determina primeiro o endereço efetivo do operando fonte, seguindo-se a determinação do endereço efetivo do operando destino.

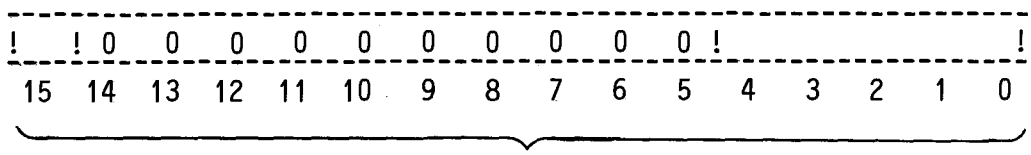
a) Formato A - Instruções com dois operandos:



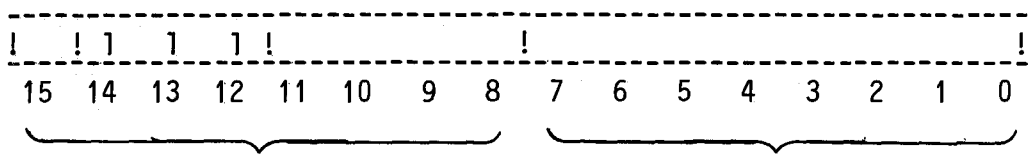
b) Formato B - Instruções com um operando:



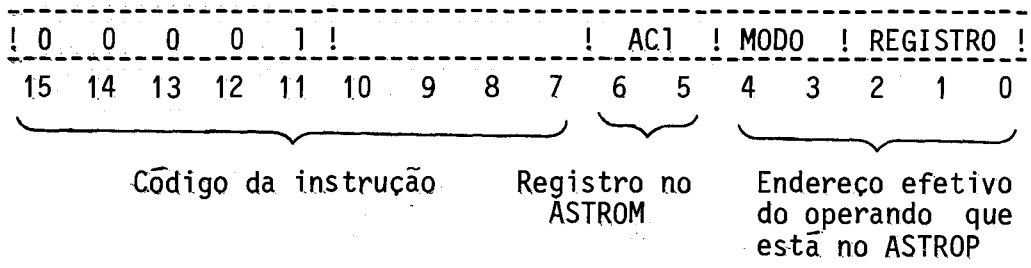
c) Formato C - Instruções com zero operandos:



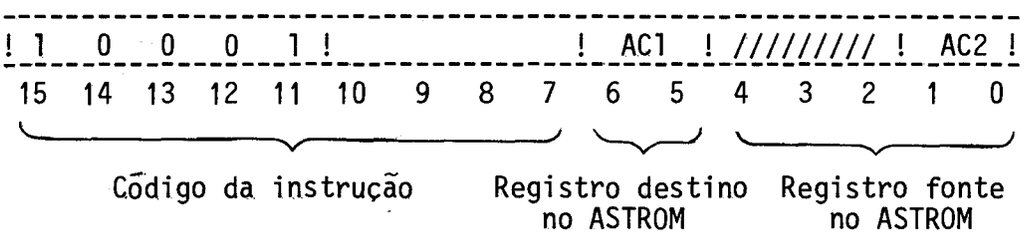
d) Formato D - Instruções com parâmetro:



e) Formato E - Instruções para unidade aritmética ASTROM:



f) Formato F - Instruções para a unidade aritmética ASTROM:



4.2 - LISTA DAS INSTRUÇÕES

As instruções do ASTROP são listadas nesta seção por classes funcionais. Além do mnemônico e do nome, consta das listas o formato - A, B, C, D, E ou F -, entre parênteses, para facilitar a localização da descrição detalhada da instrução na seção que se segue.

4.2.1 - INSTRUÇÕES ARITMÉTICAS E LÓGICAS

ADD(B) - Soma fonte ao destino (A)

SUB(B) - Subtrai fonte do destino (A)

DAC(B) - Soma fonte e bit de transporte ao destino (A)

DSB(B) - Subtrai fonte e bit de empréstimo do destino (A)

AND(B) - "E" lógico entre fonte e destino (A)

OR(B) - "OU" lógico entre fonte e destino (A)

XOR(B) - "OU-EXCLUSIVO" lógico entre fonte e destino (A)

(&) XNR(B) - "OU-EXCLUSIVO" lógico barrado entre fonte e destino (A)

(&) MSK(B) - Mascara destino com fonte (A)

NEG(B) - Negação em complemento de dois (B)

CMT(B) - Complementa (B)

STZ(B) - Armazena zeros (B)

STD(B) - Armazena uns (B)

ABS(B) - Valor absoluto (B)

ADC(B) - Soma bit de transporte ao destino (B)

SB(B) - Subtrai bit de empréstimo do destino (B)

(&) SXT(B) - Extensão do sinal (B)

INC(B) - Incrementa de um (B)

DCR(B) - Decrementa de um (B)

(&) ICT(B) - Incrementa de dois (B)

(&) DCT(B) - Decrementa de dois (B)

CMP(B) - Compara fonte com destino (A)

COC(B) - Compara os uns correspondentes entre fonte e destino (A)

(&) CZC(B) - Compara os zeros correspondentes entre fonte e destino (A)

CPZ(B) - Compara com zero (B)

CFO - Faz códigos de condição iguais a "1" (D)

CFZ - Faz códigos de condição iguais a "0" (D)

CCC - Carrega códigos de condições (D)

4.2.2 - INSTRUÇÕES DE DESLOCAMENTO E ROTAÇÃO

SLE(B) - Deslocamento para a esquerda (B)

SEC(B) - Deslocamento para esquerda com "carry" (B)

SLD(B) - Deslocamento lógico para a direita (B)

SDC(B) - Descolamento para direita com "carry" (B)

SAD(B) - Deslocamento aritmético para a direita (B)

RTE(B) - Rotação para a esquerda (B)

REC(B) - Rotação para a esquerda com "carry" (B)

RTD(B) - Rotação para a direita (B)

RDC(B) - Rotação para a direita com "carry" (B)

SLEL - Deslocamento longo para a esquerda (B)

SECL - Deslocamento longo para a esquerda com "carry" (B)

SLDL - Deslocamento lógico longo para a direita (B)

SDCL - Deslocamento longo para a direita com "carry" (B)

SADL - Deslocamento aritmético longo para a direita (B)

RTEL - Rotação longa para a esquerda (B)

RECL - Rotação longa para a esquerda com "carry" (B)

RTDL - Rotação longa para a direita (B)

RDCL - Rotação longa para a direita com "carry" (B)

4.2.3 - INSTRUÇÕES DE TRANSFERÊNCIA DE DADOS

MOV(B) - Move fonte para o destino (A)

(&) EXC(B) - Permuta fonte com destino (A)

SWB - Permuta "bytes" (B)

SRG - Salva registro de uso geral (C)

CRG - Restaura registros de uso geral (C)

CLP - Carrega limite da pilha (B)

SLP - Salva limite da pilha (B)

CSPS - Carrega ponteiro da pilha do sistema (B)

SSPS - Salva ponteiro da pilha do sistema (B)

4.2.4 - INSTRUÇÕES DE CONTROLE DO FLUXO DO PROGRAMA

JMP - Desvio incondicional (B)

DJZ - Decrementa fonte e desvia se resultado for diferente de zero (A)

JSB - Desvio para sub-rotina (A)

RET - Retorno de sub-rotina (B)

DR - Desvio relativo incondicional (D)

DCO - Desvio relativo se bit C for "1" (D)

DCZ - Desvio relativo se bit C for "0" (D)

DVO - Desvio relativo se bit V for "1" (D)

DVZ - Desvio relativo se bit V for "0" (D)

DZO - Desvio relativo se bit Z for "1" (D)

DZZ - Desvio relativo se bit Z for "0" (D)

DSO - Desvio relativo se bit S for "1" (D)

DSZ - Desvio relativo se bit S for "0" (D)

DTAO - Desvio relativo se bit T1 for "1" (D)

DTAZ - Desvio relativo se bit T1 for "0" (D)

DTBO - Desvio relativo se bit T2 for "1" (D)

DTBZ - Desvio relativo se bit T2 for "0" (D)

DTCO - Desvio relativo se bit T3 for "1" (D)

DTCZ - Desvio relativo se bit T3 for "0" (D)

DGE - Desvio relativo se maior ou igual (números com sinal)(D)

DLT - Desvio relativo se menor (números com sinal)(D)

DGT - Desvio relativo se maior (números com sinal)(D)

DLE - Desvio relativo se menor ou igual (números com sinal)(D)

DHI - Desvio relativo se maior (números sem sinal)(D)

DLS - Desvio relativo se menor ou igual (números sem sinal)(D)

DHS - Desvio relativo se maior ou igual (números sem sinal)(D)

DLO - Desvio relativo se menor (números sem sinal)(D)

ERC - Entra região condicional (D)

4.2.5 - INSTRUÇÕES DE "TRAP" E INTERRUPÇÃO

CMK - Carrega máscara de interrupção (B)

SMK - Salva máscara de interrupção (B)

MKI - Mascara algumas interrupções (B)

DMI - Desmascara algumas interrupções (B)

CNM - Carrega nível mínimo de interrupção (D)

EXS - Permuta PC/PSW com a pilha do sistema (C)

RTT - Retorno de "trap" (C)

RTI - Retorno de interrupção (C)

BTP - "Trap" de "breakpoint" (C)

TRAP(n) - Devio para "trap" n (C)

4.2.6 - OUTRAS INSTRUÇÕES

HALT - "Halt" da UCP (C)

WAIT - Espera interrupção (C)

RST - "Reset" do ASTROP (C)

NOP - Nenhuma operação (C)

4.2.7 - INSTRUÇÕES PARA A UNIDADE ARITMÉTICA ASTROM

PFIM - Converte ponto flutuante em inteiro (F)

IPFM - Converte inteiro em ponto flutuante (F)

SMPFM - Soma ponto flutuante (F)

SMIM - Soma inteiro (F)

SUPFM - Subtrai ponto flutuante (F)

SUIM - Subtrai inteiro (F)

MLPFM - Multiplica ponto flutuante (F)

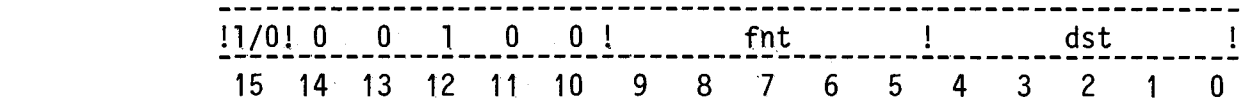
4.3 - DESCRIÇÃO DAS INSTRUÇÕES

4.3.1 - INSTRUÇÕES COM DOIS OPERANDOS

As instruções com dois operandos encontram-se descritas nesta seção na seguinte ordem:

ADD
ADDB

Soma fonte ao destino.



Operação: (dst) <-- (fnt) + (dst).

Códigos de condição: C: 1, se houver transporte do bit mais significativo do resultado;
0, Caso contrário.

V: 1, se ocorrer "overflow" aritmético, ou seja, se os operandos tiverem o mesmo sinal e o resultado apresentar sinal oposto;
0, caso contrário.

Z: 1, se o resultado der igual a zero;
0, caso contrário.

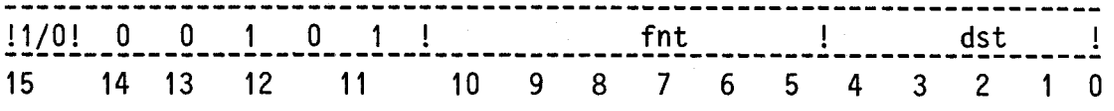
S: 1, se o resultado der menor do que zero;
0, caso contrário.

T1, T2, T3: não são afetados.

Descrição: Soma o operando fonte ao operando destino e armazena o resultado no endereço do destino. O conteúdo original do destino é perdido. O operando fonte não é alterado. A adição é feita em complemento de dois.

DAC
DACB

Soma fonte e bit de transporte ao destino.



Operação: (dst) <-- (fnt) + (dst) + (C).

Códigos de condição: C: 1, se houver transporte do bit mais significativo do resultado;

0, caso contrário.

V: 1, se ocorrer "overflow" aritmético;

0, se caso contrário.

Z: 1, se o resultado der igual a zero;

0, se caso contrário.

S: 1, se o resultado der menor do que zero;

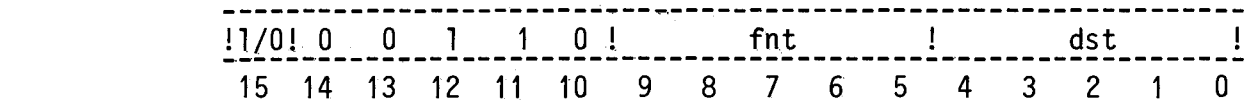
0, se caso contrário.

T1, T2, Te: não são afetados.

Descrição: Soma o operando fonte ao operando destino mais o bit de transporte C e armazena o resultado no endereço do destino. O conteúdo original do destino é perdido. O operando fonte não é alterado. A adição é feita em complemento de dois.

SUBB

Subtrai fonte do destino.



Operação: (dst) <-- (dst) - (fnt).

[em detalhe: (dst) <-- (dst) + [-](fnt) + 1]

Códigos de condição: C: 0, se houver transporte do bit mais significativo do resultado;

1, caso contrário.

V: 1, se ocorrer "overflow" aritmético, ou seja, se os operandos tiverem sinais opostos e o sinal do operando fonte for o mesmo do resultado;

0, caso contrário.

Z: 1, se o resultado der igual a zero;

0, caso contrário.

S: 1, se o resultado der menor do que zero;

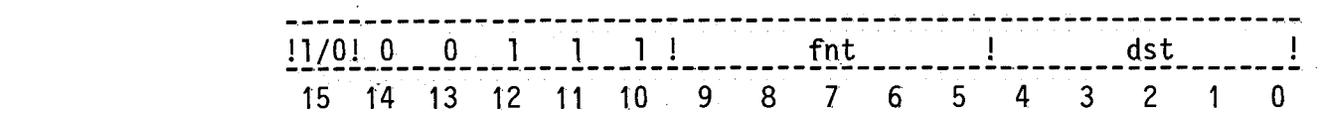
0, caso contrário.

T1, T2, T3: não são afetados.

Descrição: Subtrai o operando fonte do operando destino e deixa o resultado no endereço do destino. O conteúdo original do destino é perdido. O operando fonte não é alterado. Em aritmética de dupla precisão o código de condição C, quando feito igual a 1, indica a tomada de um bit de empréstimo ("borrow").

DSB
DSBB

Subtrai fonte e bit de empréstimo do destino.



Operação: (dst) <-- (dst) - (fnt) - (C).

[em detalhe: (dst) <-- (dst) + [-](fnt) + [-](C)]

Códigos de condição: C: 0, se houver transporte do bit mais significativo do resultado;

1, caso contrário.

V: 1, se ocorrer "overflow" aritmético;

0, caso contrário.

Z: 1, se o resultado for igual a zero;

0, caso contrário.

S: 1, se o resultado der menor do que zero;

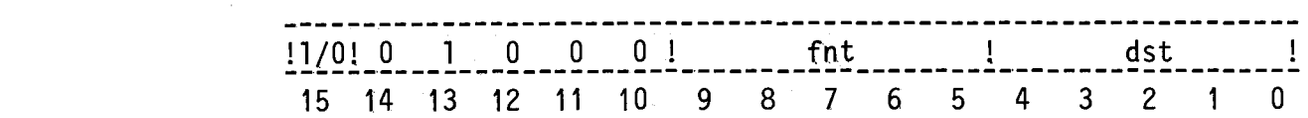
0, caso contrário.

T1, T2, T3: não são afetados.

Descrição: Subtrai o operando fonte e o bit de empréstimo do operando destino e armazena o resultado no endereço do destino. O conteúdo original do destino é perdido. O operando fonte não é alterado. Em aritmética de dupla precisão o código de condição C, quando feito igual a 1, indica a tomada de um bit de empréstimo ("borrow").

CMP
CMPB

Compara fonte com destino.



Operação: (fnt)-(dst). [em detalhe: (fnt) + [-](dst) + 1]

Códigos de condição: C: 0, se houver transporte do bit mais significativo do resultado;

1, caso contrário.

V: 1, se ocorrer "overflow" aritmético, ou seja, se os operandos tiverem sinais opostos e o sinal do operando fonte for o mesmo do resultado;

0, caso contrário.

Z: 1, se o resultado for igual a zero;

0, caso contrário.

S: 1, se o resultado for menor do que zero;

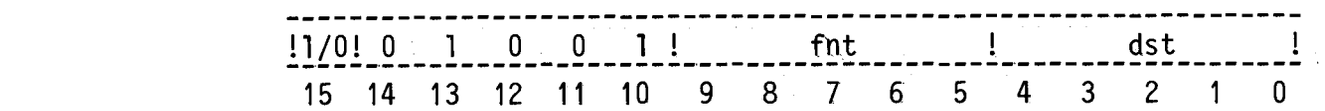
0, caso contrário.

T1, T2, T3: não são afetados.

Descrição: Compara o operando fonte com o operando destino, alterando os códigos de condição. Nenhum dos operandos é alterado. O único efeito é alterar os códigos de condição para ser usados pelas instruções de desvio condicional.

AND
ANDB

"E" lógico entre fonte e destino



Operação: (dst) <-- (fnt) [.] (dst).

Códigos de condição: C: não é afetado.

V: tornado zero.

Z: 1, se o resultado der igual a zero;
0, caso contrário.

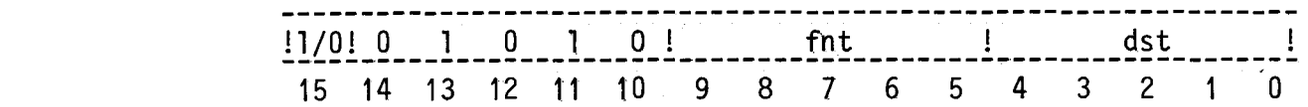
S: recebe o bit mais significativo do resultado.

T1, T2, T3: não são afetados.

Descrição: Executa um "E" lógico entre o operando fonte e o operando destino armazenando o resultado no endereço do destino, ou seja, são zerados os bits do destino que apresentam bits correspondentes iguais a zero no operando fonte. O conteúdo original do destino é perdido. O operando fonte não é alterado.

DR
DRB

"OU" lógico entre fonte e destino



Operação: (dst) <-- (fnt) [+] (dst).

Códigos de condição: C: não é afetado.

V: tornado zero.

Z: 1, se o resultado for igual a zero;
0, caso contrário.

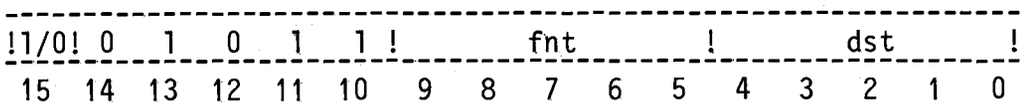
S: recebe o bit mais significativo do resultado.

T1, T2, T3: não são afetados.

Descrição: Executa um "OU" lógico entre o operando fonte e o operando destino armazenando o resultado no endereço do destino, ou seja, são tornados uns os bits do destino que apresentam bits correspondentes iguais a um no operando fonte. O conteúdo original do destino é perdido. O operando fonte não é alterado.

(&) MSK
(&) MSKB

Mascara destino com fonte



Operação: (dst) <-- [-] (fnt) [.] (dst).

Códigos de condição: C: não é afetado.

V: tornado zero.

Z: 1, se o resultado for igual a zero;
0, caso contrário.

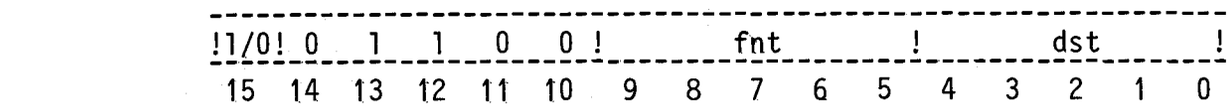
S: recebe o bit mais significativo do resultado.

T1, T2, T3: não são afetados.

Descrição: Zera cada bit do destino que apresenta bit correspondente igual a um no operando fonte. O conteúdo original do destino é perdido. O operando fonte não é alterado.

XOR
XORB

"OU-EXCLUSIVO" lógico entre fonte e destino



Operação: (dst) <-- (fnt) [*] (dst)

Códigos de condição: C: não é afetado.

V: tornado zero.

Z: 1, se o resultado for igual a zero;
0, caso contrário.

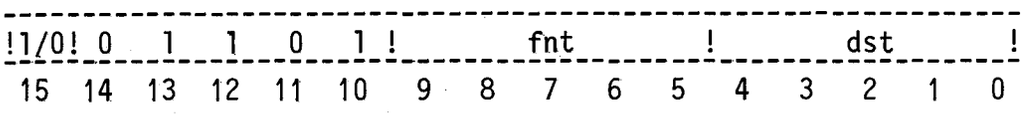
S: recebe o bit mais significativo do resultado.

T1, T2, T3: não são afetados.

Descrição: Executa um "OU-EXCLUSIVO" lógico entre o operando fonte e o operando destino armazenando o resultado no endereço do destino. O conteúdo original do destino é perdido. O operando fonte não é alterado.

(&) XNR
(&) XNRB

"OU-EXCLUSIVO" lógico barrado entre fonte e destino



Operação: (dst) <-- [-] [(fnt) [*] (dst)].

Códigos de condição: C: não é afetado.

V: tornado zero.

Z: 1, se o resultado for igual a zero:
0, caso contrário.

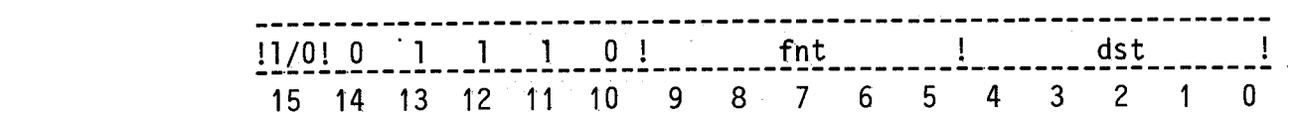
S: recebe o bit mais significativo do resultado.

T1, T2, T3: não são afetados.

Descrição: Executa um "OU-EXCLUSIVO" lógico barrado entre o operando fonte e o operando destino armazenando o resultado no endereço do destino. O conteúdo original do destino é perdido. O operando fonte não é alterado.

COC
COCB

Compara os uns correspondentes entre fonte e destino



Operação: [(fnt) [.] (dst)] [*] (fnt).

Códigos de condição: C: não é afetado.

V: tornado zero.

Z: 1, se o resultado for igual a zero;
0, caso contrário.

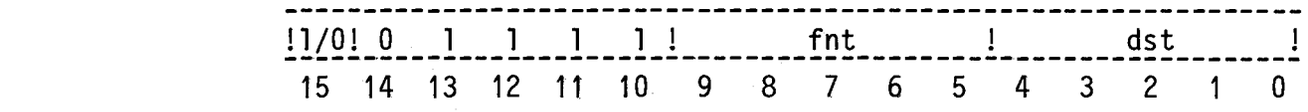
S: recebe o bit mais significativo do resultado.

T1, T2, T3: não são afetados.

Descrição: Determina se o operando destino apresenta bits uns nas mesmas posições onde existem bits uns no operando fonte. Se isto ocorrer o código de condição Z é feito igual a um, caso contrário é feito igual a zero. Nenhum dos operandos é alterado.

(&) CZC
(&) CZCB

Compara os zeros correspondente entre fonte e destino



Operação: [(fnt) [+] (dst)] [*] (fnt).

Códigos de condição: C : não é afetado.

V: tornado zero.

Z: 1, se o resultado for igual a zero;
0, caso contrário.

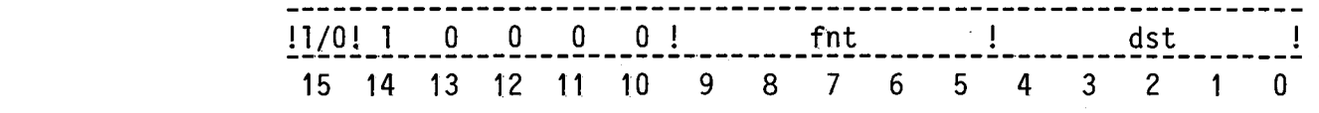
S: recebe o bit mais significativo do resultado.

T1, T2, T3: não são afetados.

Descrição: Determina se o operando destino apresenta bits zeros nas mesmas posições onde existem bits zeros no operando fonte. Se isto ocorrer o código de condição Z é feito igual a um, caso contrário é feito igual a zero. Nenhum dos operandos é alterado.

MOV
MOVB

Move fonte para destino



Operação: (dst) <-- (fnt).

Códigos de condição: C: não é afetado.

V: tornado zero.

Z: 1, se o resultado for igual a zero;
0, caso contrário.

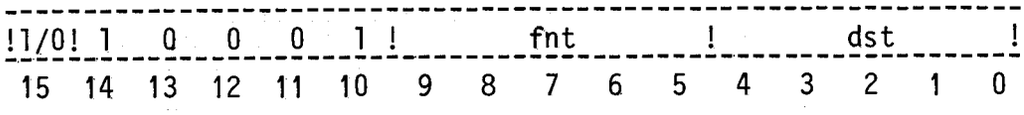
S: recebe o bit mais significativo do operando fonte.

T1, T2, T3: não são afetados.

Descrição: Move o operando fonte para o endereço do operando destino. O conteúdo original do destino é perdido. O operando fonte não é alterado.

(&) EXC
(&) EXCB

Permuta fonte com destino



Operação: (dst) <-->(fnt).

Códigos de condição: C: não é afetado.

V: tornado zero.

Z: 1, se o resultado for igual a zero;
0, caso contrário.

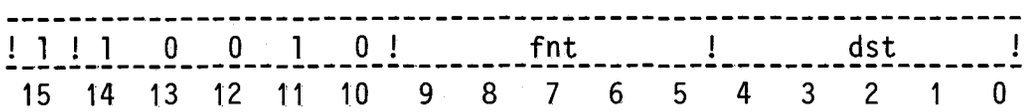
S: recebe o bit mais significativo do operando fonte.

T1, T2, T3: não são afetados.

Descrição: Permuta o conteúdo dos endereços fonte e destino. Ambos os operandos são alterados.

DJZ

Decrementa fonte e desvia se o resultado for diferente de zero



Operação: (fnt) <-- (fnt) - 1;

(PC) <-- (dst) se (fnt) for diferente de zero.

Códigos de condição: C: não é afetado.

V: 1, se o operando fonte original for o menor número negativo em complemento de dois;

0, caso contrário.

Z: 1, se o operando fonte receber o valor zero;

0, caso contrário.

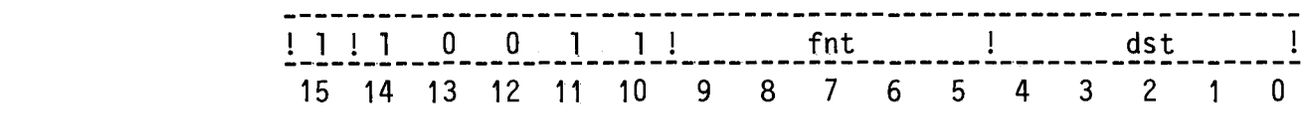
S: recebe o bit mais significativo do resultado armazenado no operando fonte.

T1, T2, T3: não são afetados.

Descrição: O operando fonte é decrementado. Se o resultado não for zero ocorre um desvio do fluxo do programa para a posição de memória definida pelo operando destino. Esta instrução provê um método eficiente de implementação de "loops". O acesso aos operandos fonte e destino são de palavra e sempre são feitos qualquer que seja o resultado da operação sobre o operando fonte.

JSB

Desvio para sub-rotina



Operação: (tmp1) <-- (fnt); [obs.: tmp1 e tmp2 são registros
(tmp2) <-- (dst); internos da UCP]

(fnt) <-- (PC);

(SP) <-- (SP) - 2; [Empilha (fnt) original]

((SP)) <-- (tmp1);

(PC) <-- (tmp2).

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

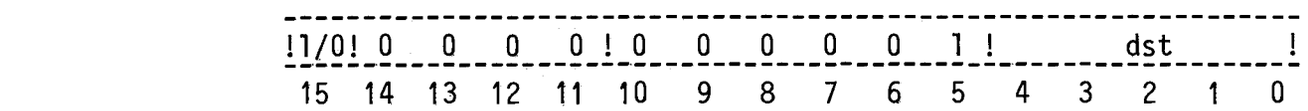
Descrição: Na execução da instrução JSB, o operando fonte é salvo na pilha de programa automaticamente e no seu lugar é colocado o endereço de retorno da sub-rotina. Com isto, sub-rotinas podem ser chamadas dentro de sub-rotinas em qualquer profundidade e usando o mesmo endereço fonte. Não há profundidade máxima preestabelecida, nem necessidade de incluir instruções dentro de cada sub-rotina para salvar e restaurar o endereço de retorno. Portanto, como todos os endereços de retorno são salvos usando a pilha de forma reentrante, a execução de uma sub-rotina pode ser interrompida e na rotina de atendimento da interrupção esta mesma sub-rotina pode ser chamada de executada. A execução inicial da sub-rotina pode ser retomada, no ponto em que estava, após a interrupção ser satisfeita. Este processo pode ocorrer até qualquer nível. Como o endereço de retorno não é salvo na pilha, é possível usar este fato, junto com os modos de endereçamento, para apontar para os parâmetros da sub-rotina que ficaram no corpo de programa principal que chama a sub-rotina em questão. Os acessos aos operandos fonte e destino são de palavra.

4.3.2 - INSTRUÇÕES COM UM OPERANDO

As instruções com um operando encontram-se descritas nesta seção na seguinte ordem:

NEG
NEGB

Negação em complemento de dois



Operação: (dst) <-- - (dst).

[em detalhe: (dst) <-- [-](dst) + 1]

Códigos de condição: C: 0, se houver transporte do bit mais significativo do resultado;

1, caso contrário.

V: 1, se o resultado for o menor número negativo em complemento de dois;

0, caso contrário.

Z: 1, se o resultado der igual a zero;

0, caso contrário.

S: 1, se o resultado der menor do que zero;

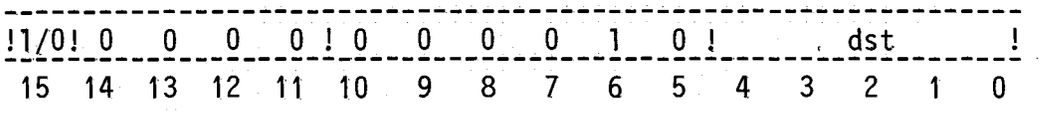
0, caso contrário.

T1, T2, T3: não são afetados.

Descrição: Coloca no destino o complemento de dois do operando destino original. Observe que "80"H em 8 bits e "8000"H em 16 bits são trocados por eles mesmos,já que em complemento de dois o número mais negativo não possui representação positiva correspondente.

INC
INCB

Incrementa de um



Operação: (dst) <-- (dst) + 1.

Códigos de condição: C: não é afetado.

V: 1, se o operando destino original for o maior número positivo em complemento de dois;

0, caso contrário.

Z: 1, se o resultado der igual a zero;

0, caso contrário.

S: 1, se o resultado der menor do que zero;

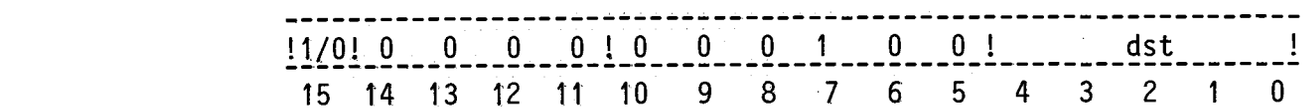
0, caso contrário.

T1, T2, T3: não são afetados.

Descrição: Soma um ao operando destino.

(&) ICT
(&) ICTB

Incrementa de dois



Operação: (dst) <-- (dst) + 2.

Códigos de condição: C: não é afetado.

V: 1, se ocorrer "overflow" aritmético;
0, caso contrário.

Z: 1, se o resultado der igual a zero;
0, caso contrário.

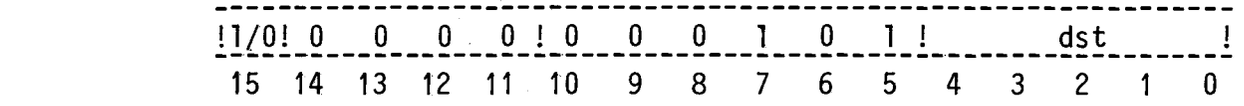
S: 1, se o reusltado der menor do que zero;
0, caso contrário.

T1, T2, T3: não são afetados.

Descrição: Soma dois ao operando destino.

(&) DCT
(&) DCTB

Decrementa de dois



Operação: (dst) <-- (dst) - 2.

Códigos de condição: C: não é afetado.

V: 1, se ocorrer "overflow" aritmético;
0, caso contrário.

Z: 1, se o resultado for igual a zero;
0, caso contrário.

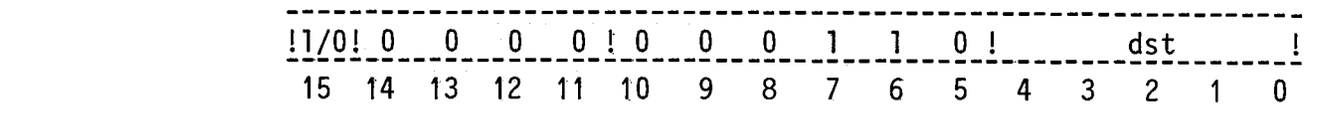
S: 1, se o resultado for menor do que zero;
0, caso contrário.

T1, T2, T3: não são afetados.

Descrição: Subtrai dois do operando destino.

CMT
CMTB

Complementa



Operação: (dst) <-- [-](dst).

Códigos de condição: C: não é afetado.

V: tornado zero.

Z: 1, se o resultado der igual a zero;
0, caso contrário.

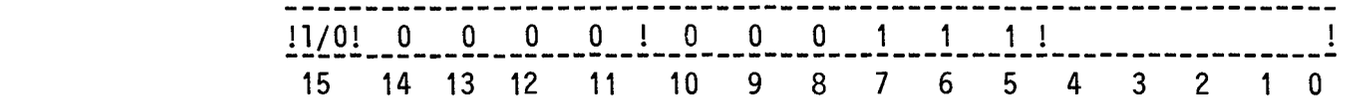
S: recebe o bit mais significativo do resultado.

T1, T2, T3: não são afetados.

Descrição: Inverte todos os bits do operando destino. Bits iguais a zero tornam-se iguais a um e bits iguais a um tornam-se iguais a zero.

STZ
STZB

Armazena zeros



Operação (dst) <-- 0.

Códigos de condição: C: não é afetado.

V: tornado zero.

Z: recebe um.

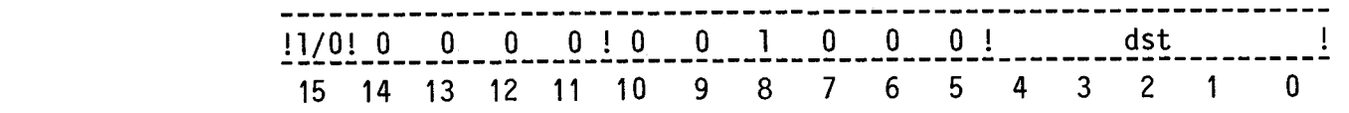
S: recebe zero.

T1, T2, T3: não são afetados.

Descrição: É colocado zero em todos os bits do destino.

STO

Armazena uns STOB



Operação: (dst) <-- 1.

Códigos de condição: C: não é afetado.

V: tornado zero.

Z: recebe zero.

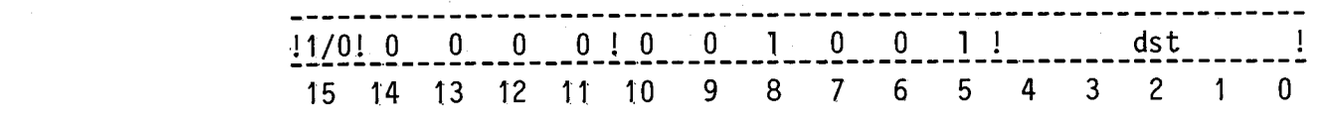
S: recebe um.

T1, T2, T3: não são afetados.

Descrição: É colocado um em todos os bits do destino.

ABS
ABSB

Valor absoluto



Operação: (dst) <-- -(dst), se (dst) < 0.

Códigos de condição: C: recebe zero.

V: 1, se o operando destino for o menor número negativo em complemento de dois;

0, caso contrário.

Z: 1, se o operando destino é zero;

0, caso contrário.

S: 1, se o operando destino for o menor número negativo em complemento de dois;

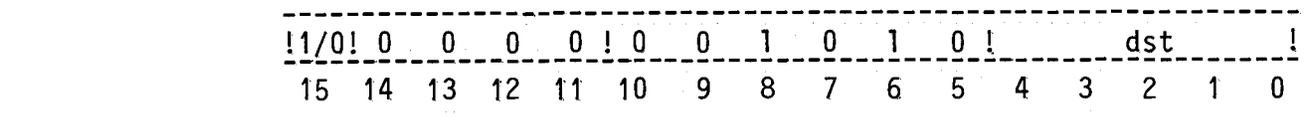
0, caso contrário.

T1, T2, T3: não são afetados.

Descrição: Coloca no destino o complemento de dois do operando destino original, se ele for negativo. Caso contrário não altera o conteúdo do destino. Observe que "80"H em 8 bits e "8000"H em 16 bits são trocados por eles mesmos já que, em complemento de dois, o número mais negativo não possui representação positiva correspondente.

ADC
ADCB

Soma bit de transporte ao destino



Operação: (dst) <-- (dst) + (C).

Códigos de condição: C: 1, se o operando destino original for menos um, em complemento de dois, e C era um;

0, caso contrário.

V: 1, se o operando destino for o maior número positivo, em complemento do dois, e C era 1;

0, caso contrário.

Z: 1, se o resultado for igual a zero;

0, caso contrário.

S: 1, se o resultado der menor do que zero;

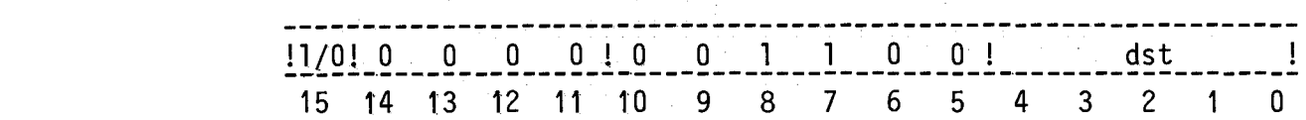
0, caso contrário.

T1, T2, T3: não são afetados.

Descrição: Soma o bit de transporte C ao operando destino.

CPZ
CPZB

Compara com zero



Operação: (dst) - 0.

Códigos de condição: C: recebe zero.

V: recebe zero.

Z: 1, se o resultado der igual a zero;
0, caso contrário.

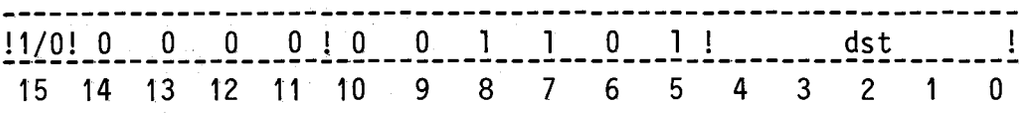
S: 1, se o resultado der menor do que zero;
0, caso contrário.

T1, T2, T3: não são afetados.

Descrição: Ativa os códigos de condição de acordo com o conteúdo do destino.

(&) SXT
(&) SXTB

Extensão do sinal



Operação: (dst) <-- 0, se o código de condição S é zero;
-1, se o código de condição S é um.

Códigos de condição: C: não é afetado.

V: tornado zero.

Z: 1, se o código de condição S é zero;
0, caso contrário.

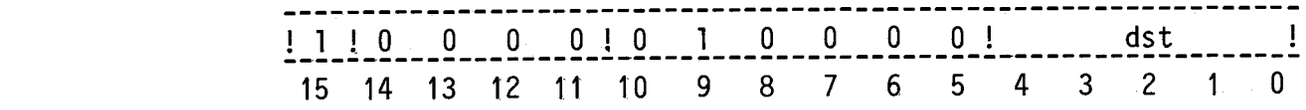
S: não é afetado.

T1, T2, T3: não são afetados.

Descrição: Coloca -1, em complemento de dois, no destino, se o código de condição S é um; se S é zero, então coloca zero no destino.

SLDL

Deslocamento lógico longo para a direita



Operação: (dst) <-- (dst) deslocado de um bit para a direita.

Código de condição: C: não é afetado.

V: tornado zero.

Z: 1, se a palavra mais significativa do resultado der igual a zero;

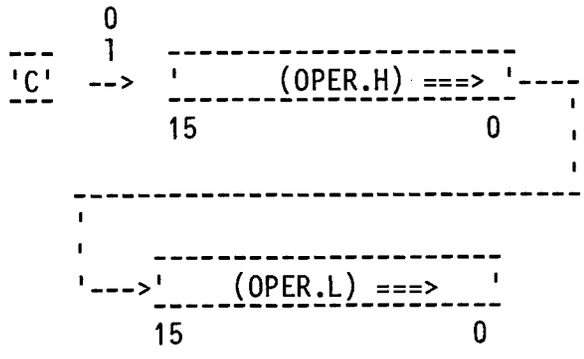
0, caso contrário.

S: recebe o bit mais significativo da palavra mais significativa do resultado.

T1, T2, T3: não são afetados.

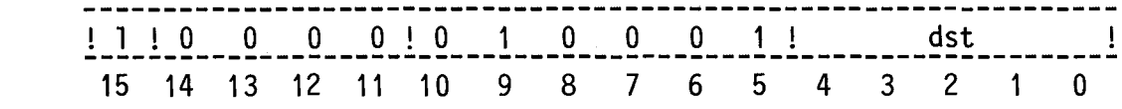
Descrição: Desloca todos os bits do operando destino, que é uma palavra longa de 32 bits, de um bit para a direita. O bit mais significativo da palavra mais significativa (OPER.H) é carregado com zero. O bit menos significativo da palavra menos significativa (OPER.L) do operando destino original é perdido.

Palavra
Longa:



SDCL

Deslocamento longo para a direita com "carry"



Operação: (dst) <-- (dst) deslocado de um bit para a direita.

Códigos de condição: C: tornado zero.

V: tornado zero.

Z: 1, se a palavra mais significativa do resultado der igual a zero;

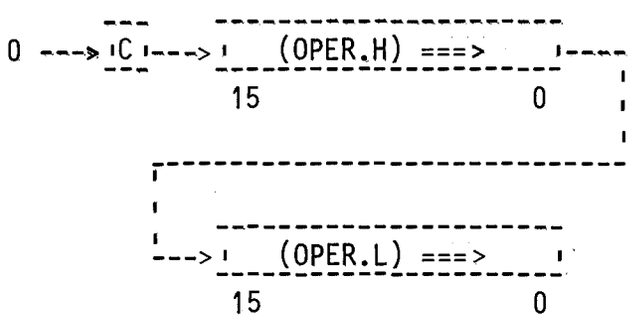
0, caso contrário.

S: recebe o bit mais significativo da palavra mais significativa do resultado.

T1, T2, T3: não são afetados.

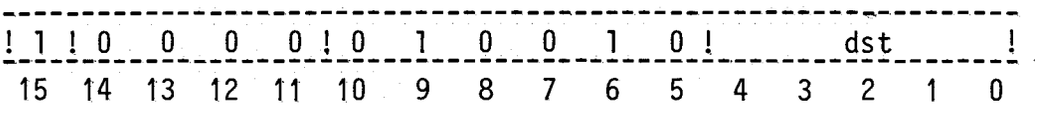
Descrição: Desloca todos os bits do operando destino, que é uma palavra longa de 32 bits, de um bit para a direita. O bit mais significativo da palavra mais significativa (OPER.H) é carregado com o código de condição C. O bit menos significativo da palavra menos significativa (OPER.L) do operando destino original é perdido. O código de condição C recebe 0.

Palavra
Longa:



RDCL

Rotação longa para direita com "carry"



Operação: (dst) <-- (dst) deslocado de um bit para a direita.

Códigos de condição: C: recebe o bit menos significativo da palavra menos significativa do operando destino.

V: tornado zero.

Z: 1, se a palavra mais significativa do resultado der igual a zero;

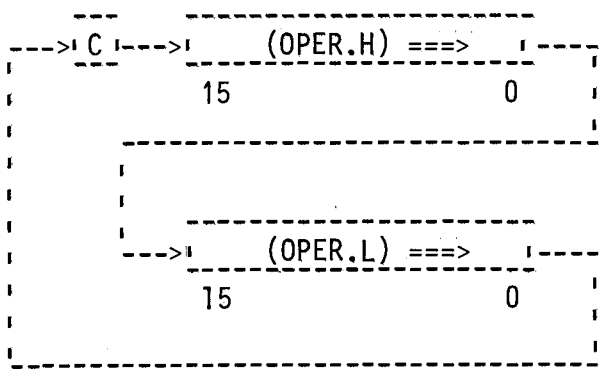
0, caso contrário.

S: recebe o bit mais significativo da palavra mais significativa do resultado.

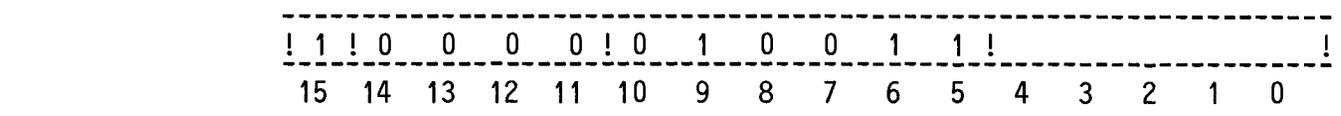
T1, T2, T3: não são afetados.

Descrição: Desloca todos os bits do operando destino que é uma palavra longa de 32 bits, de um bit para a direita. O bit mais significativo da palavra mais significativa (OPER.H) é carregado com o código de condição C que, por sua vez, recebe o bit menos significativo da palavra menos significativa (OPER.L) do operando destino original.

Palavra
longa:



Deslocamento aritmético longo para a direita



Operação: (dst) <-- (dst) deslocado de um bit para a direita.

Códigos de condiçaõ: C: não é afetado.

V: tornado zero.

Z: 1, se a palavra mais significativa do resultado der igual a zero;

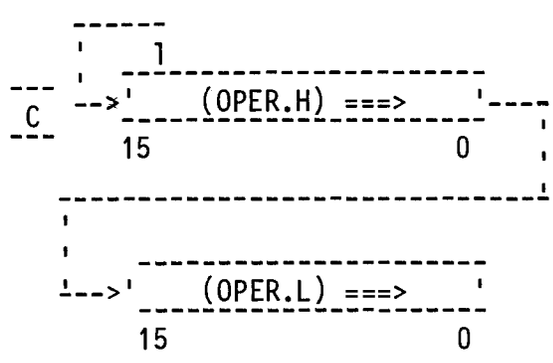
0, caso contrário.

S: recebe o bit mais significativo da palavra mais significativa do resultado.

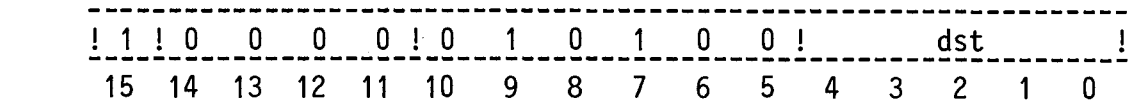
T1, T2, T3: não são afetados.

Descrição: Desloca todos os bits do operando destino, que é uma palavra longa de 32 bits, de um bit para a direita. O bit mais significativo da palavra mais significativa (OPER.H) é carregado com o bit mais significativo da palavra mais significativa do operando destino original. O bit menos significativo da palavra menos significativa (OPER.L) do operando destino original é perdido.

Palavra
Longa:



Rotação longa para a direita



Operação: (dst) <-- (dst) deslocado de um bit para a direita.

Códigos de condição: C: não é afetado.

V: tornado zero.

Z: 1, se a palavra mais significativa do resultado for igual a zero;

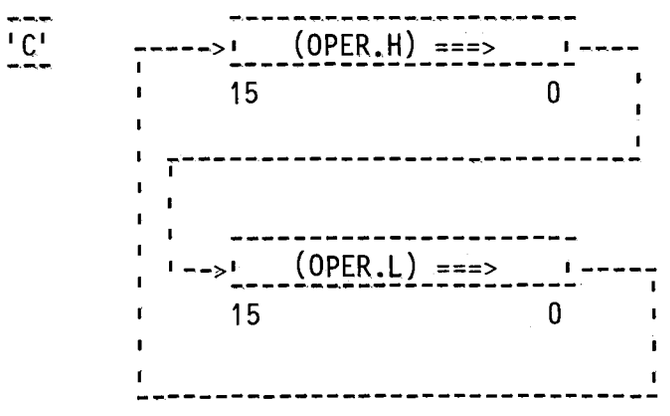
0, caso contrário.

S: recebe o bit mais significativo da palavra mais significativa do resultado.

T1, T2, T3: não são afetados.

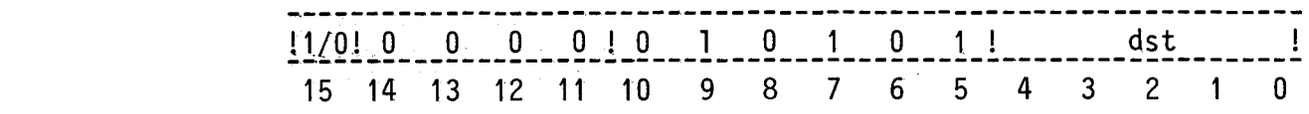
Descrição: Desloca todos os bits do operando destino, que é uma palavra longa de 32 bits, de um bit para a direita. O bit mais significativo da palavra mais significativa (OPER.H) é carregado com o bit menos significativo da palavra menos significativa (OPER.L) do operando destino original.

Palavra longa:



SLD
SLDB

Deslocamento lógico para a direita



Operação: (dst) <-- (dst) deslocado de um bit para a direita.

Códigos de condição: C: não é afetado.

V: tornado zero.

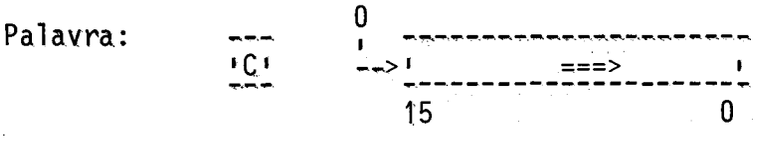
Z: 1, se o resultado der igual a zero;
0, caso contrário.

S: recebe o bit mais significativo do resultado.

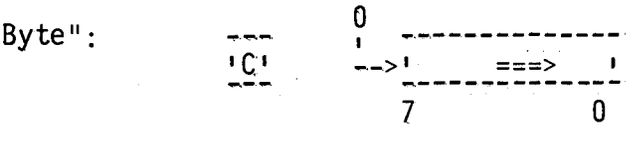
T1, T2, T3: não são afetados.

Descrição: Desloca todos os bits do operando destino de um bit para a direita. O bit mais significativo é carregado com 0. O bit menos significativo do operando destino original é perdido.

Palavra:

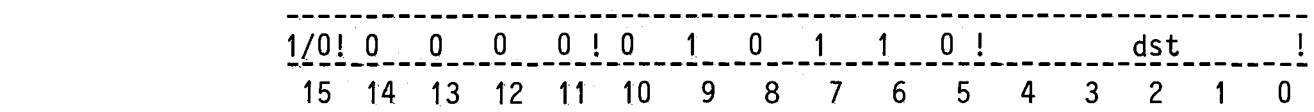


"Byte":



SDC
SDCB

Deslocamento para a direita com "carry"



Operação: (dst) <-- (dst) deslocado de um bit para a direita.

Códigos de condição: C: tornado zero.

V: tornado zero.

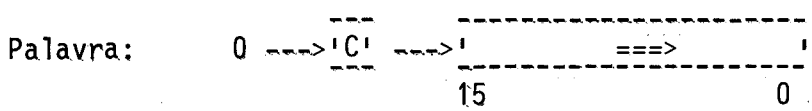
Z: 1, se o resultado for igual a zero;

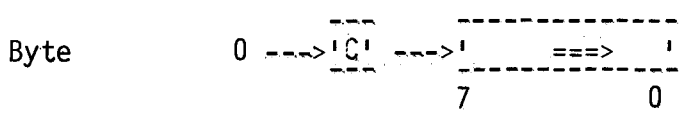
0, caso contrário.

S: recebe o bit mais significativo do resultado.

T1, T2, T3: não são afetados.

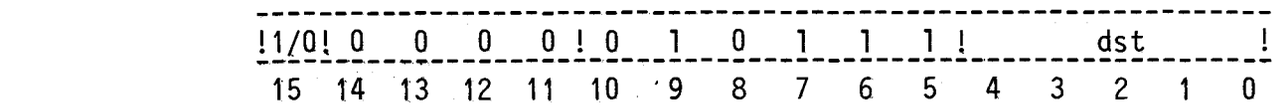
Descrição: Desloca todos os bits do operando destino de um bit para a direita. O bit mais significativo é carregado com o código de condição C. O bit menos significativo do operando destino original é perdido. O código de condição C recebe 0.





RDC
RDCB

Rotação para a direita com "carry"



Operação: (dst) <-- (dst) deslocado de um bit para a direita.

Códigos de condição: C: recebe o bit menos significativo do operando destino.

V: tornado zero.

Z: 1, se o resultado der igual a zero;

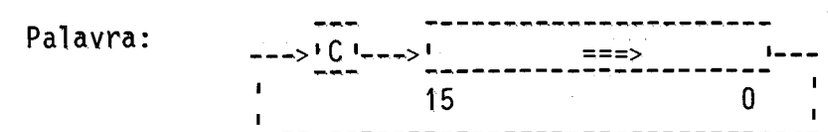
0, caso contrário.

S: recebe o bit mais significativo do resultado significativo do resultado.

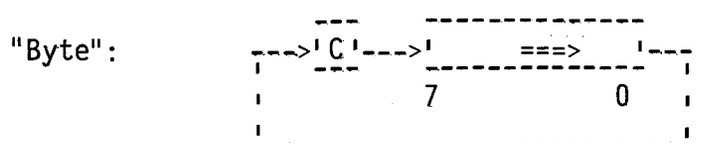
T1, T2, T3: não são afetados.

Descrição: Desloca todos os bits do operando destino de um bit para a direita. O bit mais significativo é carregado com o código de condição C que, por sua vez, recebe o bit menos significativo do operando destino original.

Palavra:

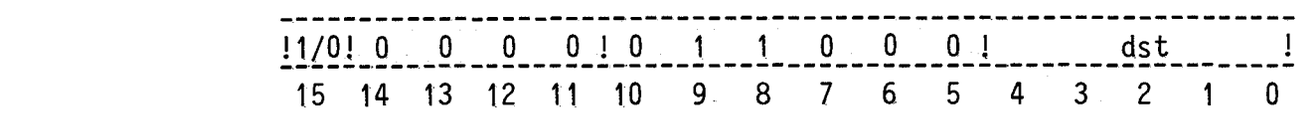


"Byte":



SAD
SADB

Deslocamento aritmético para a direita



Operação: (dst) <-- (dst) deslocado de um bit para a direita.

Códigos de condição: C: não é afetado.

V: tornado zero.

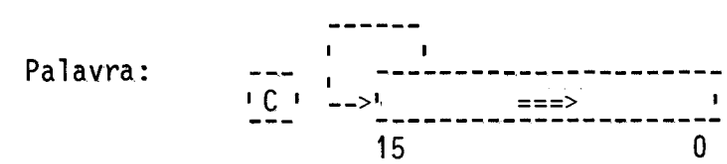
Z: 1, se o resultado der igual a zero;
0, caso contrário.

S: recebe o bit mais significativo do resultado.

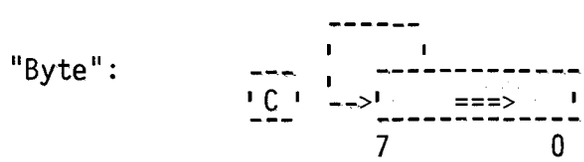
T1, T2, T3: não são afetados.

Descrição: Desloca todos os bits do operando destino de um bit para a direita. O bit mais significativo é carregado com o bit mais significativo do operando destino original. O bit menos significativo do operando destino original é perdido.

Palavra:

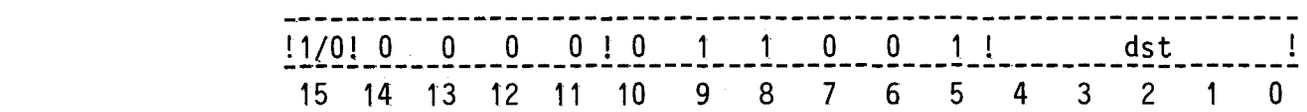


"Byte":



RTD
RTDB

Rotação para a direita



Operação: (dst) <-- (dst) deslocado de um bit para a direita.

Códigos de condição: C: não é afetado.

V: tornado zero.

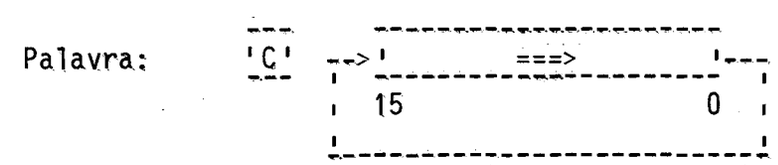
Z: 1, se o resultado der igual a zero;
0, caso contrário.

S: recebe o bit mais significativo do resultado.

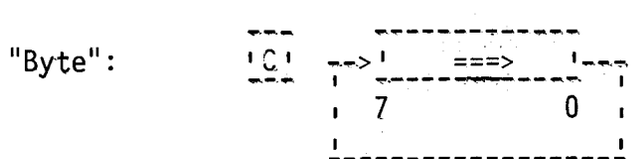
T1, T2, T3: não são afetados.

Descrição: Desloca todos os bits do operando destino de um bit para a direita. O bit mais significativo é carregado com o bit menos significativo do operando destino original.

Palavra:

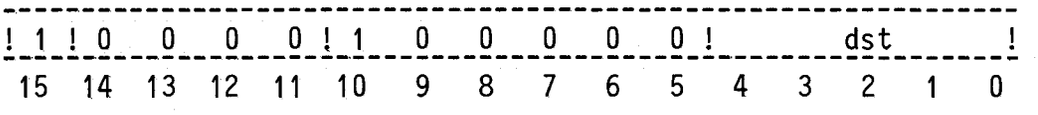


"Byte":



SLEL

Deslocamento longo para a esquerda



Operação: (dst) <-- (dst) deslocado de um bit para a esquerda.

Códigos de condição: C: não é afetado.

V: tornado zero.

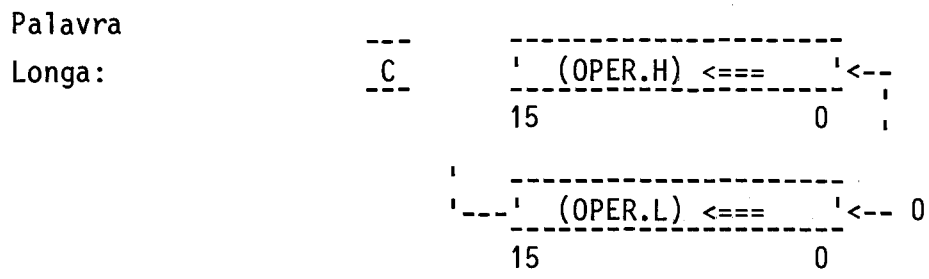
Z: 1, se a palavra mais significativa do resultado for igual a zero;

0, caso contrário.

S: recebe o bit mais significativo da palavra mais significativa do resultado.

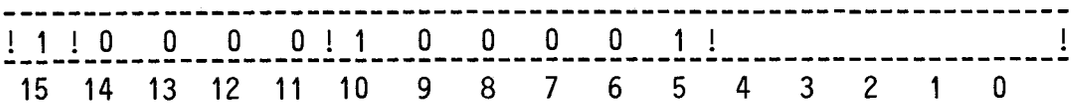
T1, T2, T3: não são afetados.

Descrição: Desloca todos os bits do operando destino, que é uma palavra longa de 32 bits, de um bit para a esquerda. O bit menos significativo da palavra menos significativa (OPER.L) é carregado com 0. O bit mais significativo da palavra mais significativa (OPER.H) do operando destino original é perdido.


Palavra
Longa:
C
(OPER.H) <===
15
0
(OPER.L) <===
15
0
0

SECL

Deslocamento longo para a esquerda com "carry"



Operação: (dst) <-- (dst) deslocado de um bit para a esquerda.

Códigos de condição: C: recebe o bit mais significativo da palavra mais significativa do operando destino.

V: tornado zero.

Z: 1, se a palavra mais significativa do resultado for igual a zero;

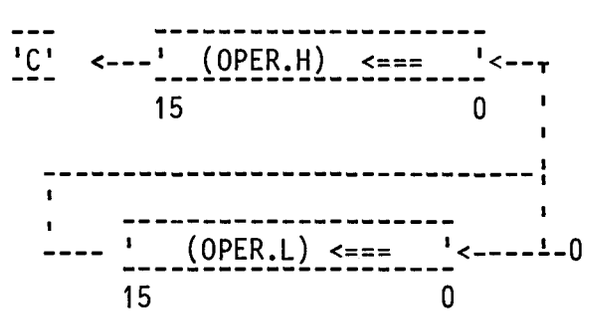
0, caso contrário.

S: recebe o bit mais significativo da palavra mais significativa do resultado.

T1, T2, T3: não são afetados.

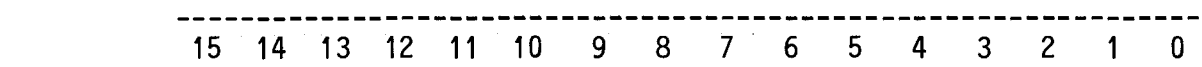
Descrição: Desloca todos os bits do operando destino, que é uma palavra longa de 32 bits, de um bit para a esquerda. O bit menos significativo da palavra menos significativa (OPER.L) é carregado com 0. O código de condição C recebe o bit mais significativo (OPER.H) do operando destino original.

Palavra Longa:



RECL

Rotação longa para a esquerda com "carry"



Operação: (dst) <-- (dst) deslocado de um bit para a esquerda.

Códigos de condição: C: recebe o bit mais significativo da palavra mais significativa do operando destino.

V: tornado zero.

Z: 1, se a palavra mais significativa do resultado for igual a zero;

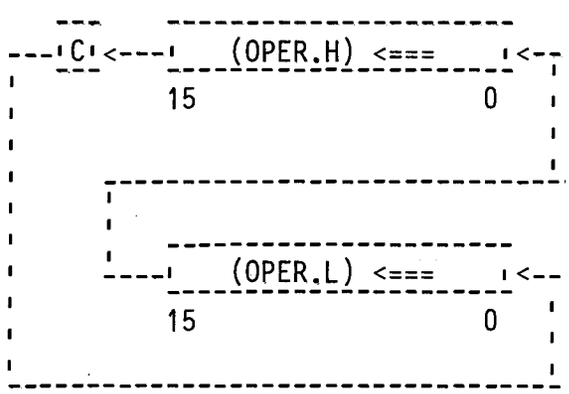
0, caso contrário.

S: recebe o bit mais significativo da palavra mais significativa do resultado.

T1, T2, T3: não são afetados.

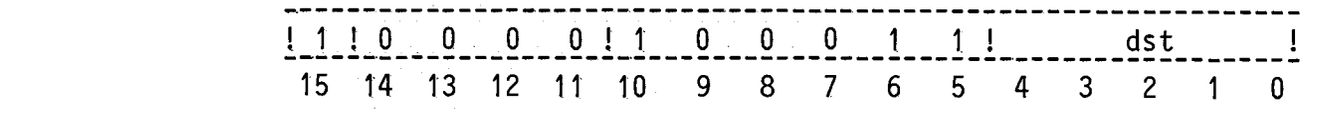
Descrição: Desloca todos os bits do operando destino, que é uma palavra longa de 32 bits, de um bit para a esquerda. O bit menos significativo da palavra menos significativa (OPER.L) é carregado com o código de condição C que, por sua vez, recebe o bit mais significativo da palavra mais significativa (OPER.H) do operando destino original.

Palavra longa:



RTEL

Rotação longa para a esquerda



Operação: (dst) <-- (dst) deslocado de um bit para a esquerda.

Códigos de condição: C: não é afetado.

V: tornado zero.

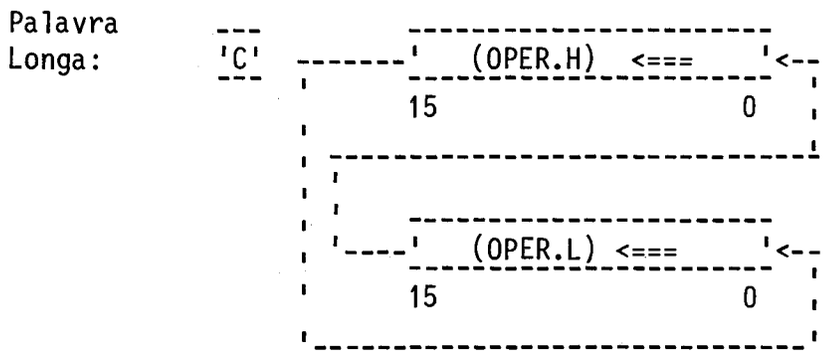
Z: 1, se a palavra mais significativa do resultado for igual a zero;

0, caso contrário.

S: recebe o bit mais significativo da palavra mais significativa do resultado.

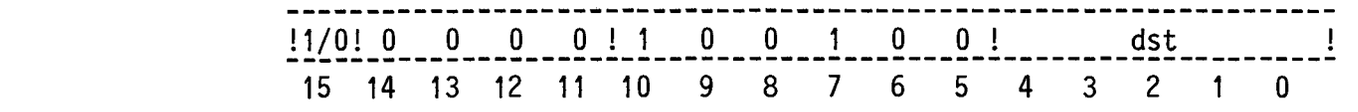
T1, T2, T3: não são afetados.

Descrição: Desloca todos os bits do operando destino, que é uma palavra longa de 32 bits, de um bit para a esquerda. O bit menos significativo da palavra menos significativa (OPER.L) recebe o bit mais significativo da palavra mais significativa (OPER.H) do operando destino original.


Palavra
Longa:
'C'
(OPER.H) <===
15
0
(OPER.L) <===
15
0

SLE
SLEB

Deslocamento para a esquerda



Operação: (dst) <-- (dst) deslocado de um bit para a esquerda.

Códigos de condição: C: não é afetado.

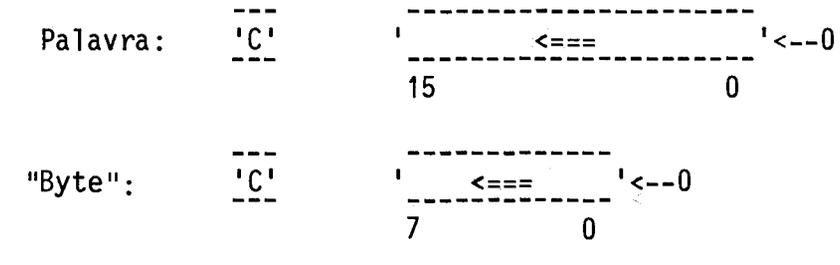
V: tornado zero.

Z: 1, se resultado der igual a zero;
0, caso contrário.

S: recebe o bit mais significativo do resultado.

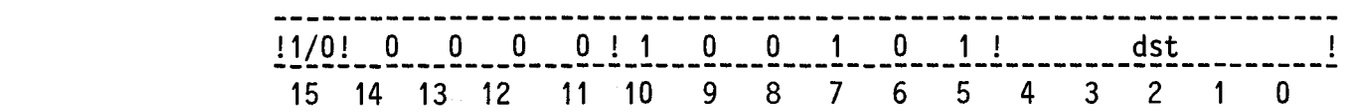
T1, T2, T3: não são afetados.

Descrição: Desloca todos os bits do operando destino de um bit para a esquerda. O bit menos significativo é carregado com 0. O bit menos significativo é carregado com 0. O bit mais significativo do operando destino original é perdido.



SEC
SECB

Deslocamento para a esquerda com "carry"



Operação: (dst) <-- (dst) deslocamento de um bit para a esquerda.

Códigos de condição: C: recebe o bit mais significativo do operando destino.

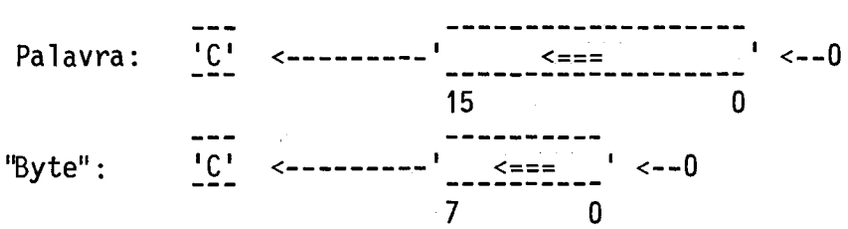
V: tornado zero.

Z: 1, se resultado for igual a zero;
0, caso contrário.

S: recebe o bit mais significativo do resultado.

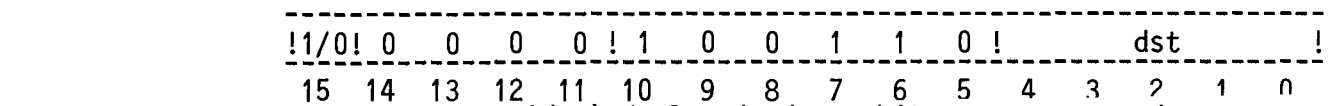
T1, T2, T3: não são afetados.

Descrição: Desloca todos os bits do operando destino de um bit para a esquerda. O bit menos significativo é carregado com 0. O código de condição C recebe o bit mais significativo do operando destino original.



REC
RECB

Rotação para a esquerda com "carry"



Operaçao: (dst) <-- (dst) deslocado de um bit para a esquerda.

Códigos de condição: C: recebe o bit mais significativo do operando destino.

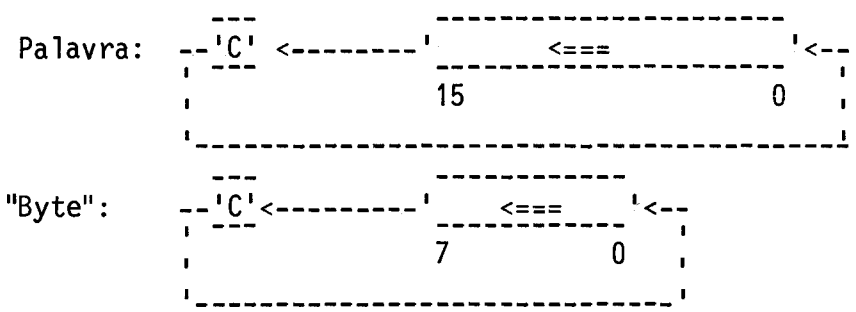
V: tornado zero.

Z: 1, se resultado der igual a zero;
0, caso contrário.

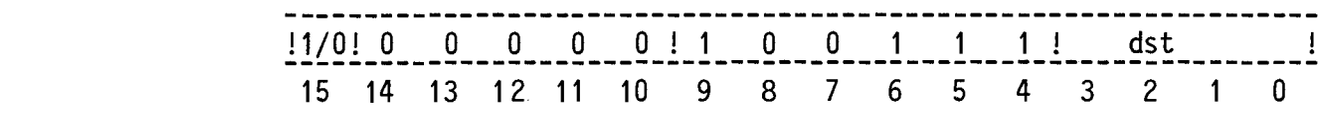
S: recebe o bit mais significativo do resultado.

T1, T2, T3: não são afetados.

Descrição: Desloca todos os bits do operando destino de um bit para a esquerda. O bit menos significativo é carregado com o código de condição C que, por sua vez, recebe o bit mais significativo do operando destino original.



Rotação para a esquerda



Operação: (dst) <-- (dst) deslocado de um bit para a esquerda.

Códigos de condição: C: não é afetado.

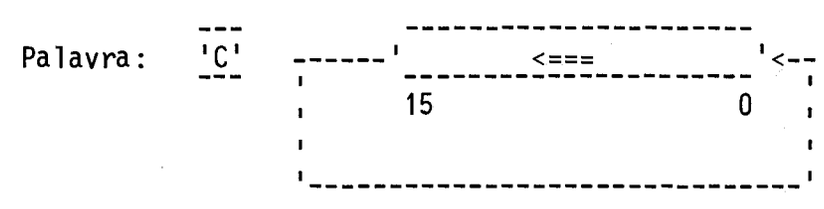
V: tornado zero.

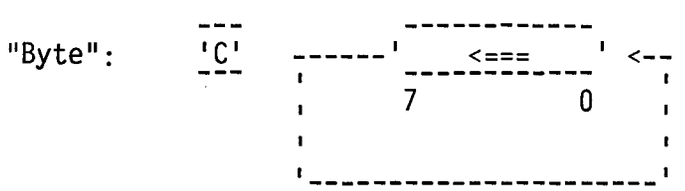
Z: 1, se resultado for igual a zero;
0, caso contrário.

S: recebe o bit mais significativo do resultado.

T1, T2, T3: não são afetados.

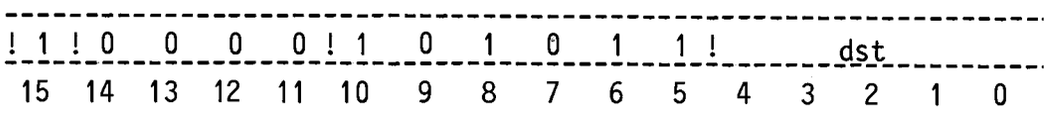
Descrição: Desloca todos os bits do operando destino de um bit para a esquerda. O bit menos significativo é carregado com o bit mais significativo do operando destino original.





SWB

Permuta "bytes"



Operação: "byte" H <--> "byte" L.

Códigos de condição: C: não é afetado.

V: tornado zero.

Z: 1, se a palavra resultante for igual a zero.
0. caso contrário.

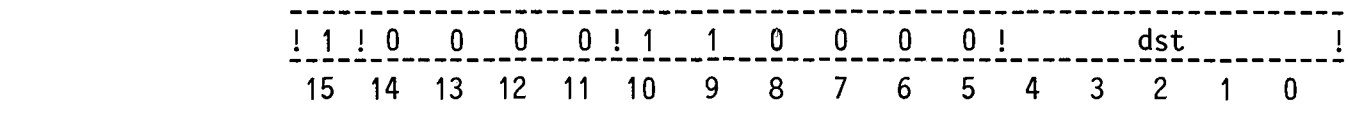
S: recebe o bit mais significativo da palavra resultante.

T1, T2, T3: não são afetados.

Descrição: Permuta o "byte" mais significativo ("byte" H) com o "byte" menos significativo ("byte" L) do operando destino que, necessariamente, tem de ser uma palavra. O acesso ao operando destino é de palavra.

RET

Retorno de sub-rotina.



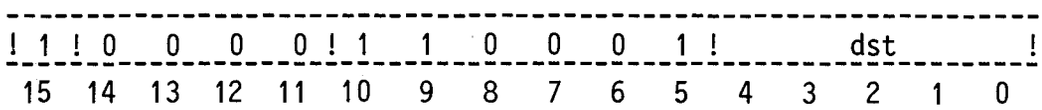
Operação: (PC) <-- (dst);
(dst) <-- ((SP)); [Desempilha]
(SP) <-- (SP) + 2.

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Carrega o registro PC com o operando destino e restaura o valor original do operando destino que havia sido salvo na pilha. O acesso ao operando destino é de palavra.

JMP

Desvio incondicional



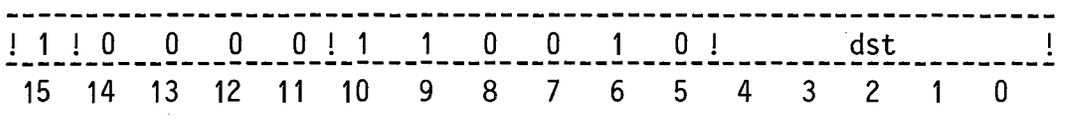
Operação: (PC) <-- (dst).

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: A instrução JMP provê a maneira mais flexível de desvio do fluxo seqüencial de execução de um programa, com o controle podendo ser transferido para qualquer localização na memória. Recomenda-se que o operando seja um endereço par, já que o bit menos significativo do registro PC está sempre preso em zero. O acesso ao operando destino é de palavra.

SLP

Salva limite da pilha



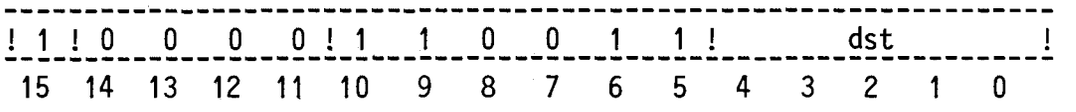
Operação: (dst) <-- (LP).

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Armazena no destino o conteúdo do registro LP (limite da pilha). O acesso ao operando destino é de palavra.

CLP

Carrega limite da pilha



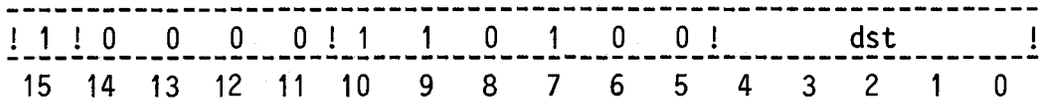
Operação: (LP) <-- (dst).

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Armazena no registro LP o operando destino. O acesso ao operando destino é de palavra.

SMK

Salva máscara de interrupção



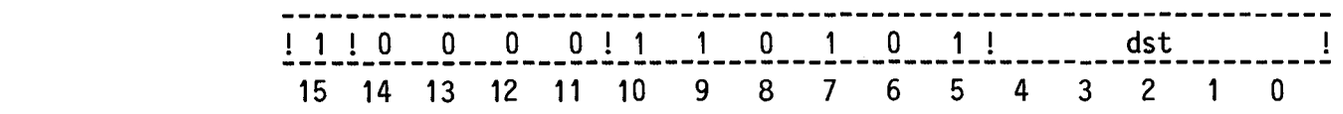
Operação: (dst) <-- (MK).

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Armazena no destino o conteúdo do registro MK (máscara de interrupção). O acesso ao operando destino é de palavra.

CMK

Carrega máscara de interrupção



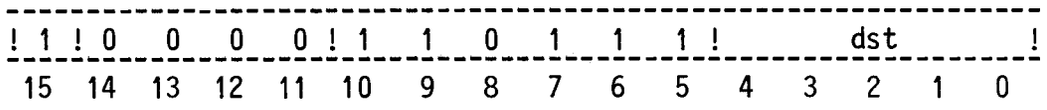
Operação: (MK) <-- (dst).

Códigos de condição: C, V. Z, S, T1, T2, T3: não são afetados.

Descrição: Armazena no registro MK (máscara de interrupção) o operando destino. O acesso ao operando destino é de palavra. O bit mais significativo do operando destino é irrelevante, já que o registro MK possui o bit 15 preso em zero.

MKI

Mascara algumas interrupções



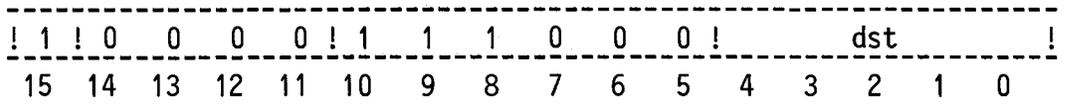
Operação: (MK) <-- (MK) [+] (dst), exceto bit 15.

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: São feitos iguais a um todos os bits do registro MK (exceto o bit 15) que possuem os bits correspondentes no operando destino iguais a um. Os outros bits não são afetados. O bit 15 do registro MK é sempre zero. O acesso ao operando destino é de palavra.

CSPS

Carrega ponteiro da pilha do sistema



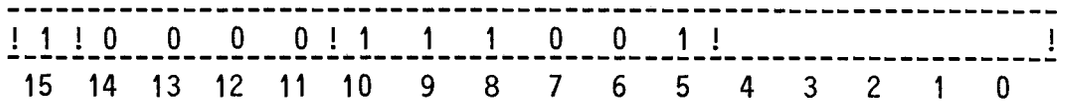
Operação: (SPS) <-- (dst).

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Armazena no registro SPS (ponteiro da pilha do sistema) o operando destino. O acesso ao operando destino é de palavra.

SSPS

Salva ponteiro da pilha do sistema



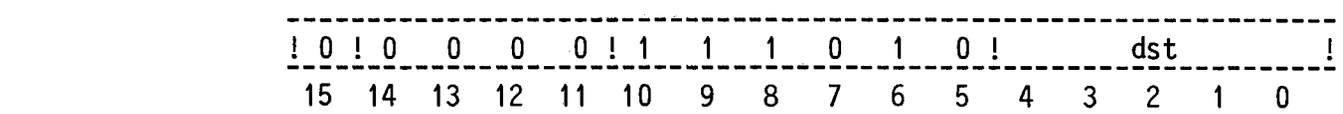
Operação: (dst) <-- (SPS).

Código de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Armazena no destino o conteúdo do registro SPS (ponteiro da pilha do sistema). O acesso ao operando destino é de palavra.

SFGMB

Salva códigos de condição do ASTROM



Operação: (dst) <-- Códigos de condição do ASTROM.

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Armazena no destino os códigos de condição do ASTROM. O acesso ao operando destino é de "byte". O novo conteúdo de destino passa a ser os seguintes códigos de condição (Seção 4.3.5):

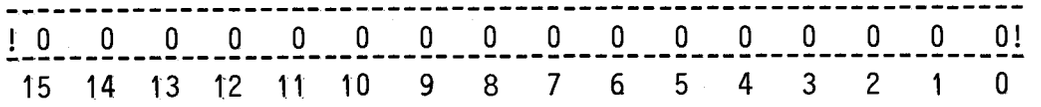
bit 0: UPFOK,
bit 1: ZUPF,
bit 2: SUPF,
bit 3: DIVZ,
bit 4: OVFL,
bit 5: UNDF,
bit 6: CVER,
bit 7: TRP* que é o OU-lógico barrado dos códigos de condição CVER, UNDF, OVFL e DIVZ.

4.3.3 - INSTRUÇÕES COM ZERO OPERANDOS

As instruções com zero operandos encontram-se descritas nesta seção na seguinte ordem:

HALT

"Halt" da UCP



Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Leva a UCP/ASTROP para o estado de "HALT", onde ela pára a execução de instruções dos programas, o que é indicado pelos LED'S CONTROLE: UCP/PAINEL do painel do computador ASTROP. Nesta situação, usando os controles e indicadores do painel, é possível o acesso aos registros internos da UCP/ASTROP e à memória do ASTROP (via BASIS), além da execução de programas passo a passo. A UCP/ASTROP apenas retornará para o estado de "RUN" se o "push-button" EXECUTE for pressionado com as chaves PASSO A PASSO/LIVRE E MODO PAINEL, ambas na posição LIVRE (estes controles e indicadores estão no painel). Após a execução da instrução HALT, a UCP/ASTROP pára com o registro PC apontando para a próxima instrução que seria executada.

WAIT

Espera interrupção

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Provê uma maneira da UCP/ASTROP abrir mão do uso do BASIS, enquanto espera por uma interrupção externa. Após executar a instrução WAIT, a UCP/ASTROP não mais competirá pelo uso do BASIS para buscar instruções e operandos na memória. Isto permite uma alta taxa de transferência entre um periférico ea memória. Na instrução WAIT, bem como durante a execução de todas as outras instruções, o registro PC aponta para a instrução seguinte. Portanto, quando uma interrupção causar o salvamento do conteúdo dos registros PC e PSW na pilha do sistema, o endereço da instrução que segue a instrução WAIT é salvo. A saída da rotina de atendimento de interrupção (ou seja, a execução da instrução RTI) causará o retorno do programa interrompido a partir da instrução que segue a instrução WAIT.

NOP

Nenhuma operação

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Nenhuma operação útil é realizada.

RST

"Reset" do ASTROP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Causa a ativação do sinal RESET do BASIS durante um segundo. Todos os dispositovos conectados ao BASIS (inclusive a UCP/ASTROP) são levados para o estado inicial de "power-up".

EXS

Permuta PC/PSW com pilha do sistema

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

```
Operação: (temp1) <-- ((SPS));   [Desempilha novo PSW]
            (SPS) <-- (SPS)+2;
          [temp2] <-- ((SPS));   [Desempilha novo PC]
            (SPS) <-- (SPS)+2;
            [SPS] <-- (SPS)-2;   [Empilha PC atual]
          ((SPS)) <-- (PC);
            (SPS) <-- (SPS)-2;   [Empilha PSW atual]
          ((SPS)) <-- (PSW);
            (PSW) <-- (temp1);
             (PC) <-- (temp2).
```

Códigos de condição: C, V, Z, S, T1, T2, T3 recebem um novo contexto.

Descrição: Permuta com o topo da pilha do sistema os conteúdos dos registros PC e PSW. Após a execução da instrução EXS, se o bit B da PSW ficar ativo, um "trap" de "breakpoint" só ocorrerá no final da execução da primeira instrução.

Retorno de "trap"

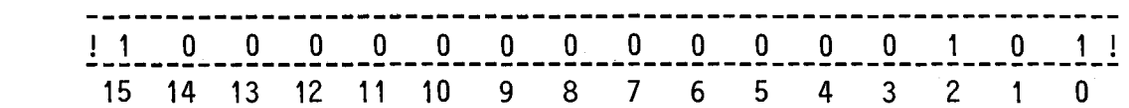
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Operação: (PSW) <-- ((SPS)); [Desempilha novo PSW]
(SPS) <-- (SPS) + 2;
(PC) <-- ((SPS)); [Desempilha novo PC]
(SPS) <-- (SPS) + 2.

Códigos de condição: C, V, Z, S, T1, T2, T3 recebem um novo contexto.

Descrição: A instrução RTT é usada para retornar de uma rotina de atendimento de "trap". Após a execução da instrução RTT, se o bit B da PSW ficar ativo, um "trap" de "breakpoint" só ocorrerá no final de execução da primeira instrução.

Retorno de interrupção



Operação: (PSW) <-- ((SPS)); [Desempilha novo PSW]
(SPS) <-- (SPS) + 2;
(PC) <-- ((SPS)); [Desempilha novo PC]
(SPS) <-- (SPS) + 2.

Códigos de condição: C, V, Z, S, T1, T2, T3 recebem um novo contexto.

Descrição: A instrução RTI, análoga a RTT, é usada para retornar de uma rotina de atendimento de interrupção. Após a execução da instrução RTI, se o bit B da PSW ficar ativo, um "trap" de "breakpoint" só ocorrerá no final de execução da primeira instrução.

BTP

"Trap" de "breakpoint"

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Operação:
```
   (SPS) <-- (SPS) - 2;   [Empilha PC atual]
((SPS)) <-- (PC);
   (SPS) <-- (SPS) - 2;   [Empilha PSW atual]
((SPS)) <-- (PSW);
   (PSW) <-- ("FF98"H);
    (PC) <-- ("FF 9A"H).
```

Códigos de condição: C, V, Z, S, T1, T2, T3 recebem um novo contexto.

Descrição: Salva na pilha do sistema o PC e a PSW, e obtém o vetor de "Trap" de "breakpoint" dos endereços "FF98"H e "FF9A"H da memória. Esta instrução é usada na depuração de programas.

SRG

Salva registro de uso geral

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Operação:
```
(SP) <-- (SP) - 2;   [Empilha RF]
((SP)) <-- (RF);
(SP) <-- (SP) - 2;   [Empilha RE]
((SP)) <-- (RE);
(SP) <-- (SP) - 2;   [Empilha RD]
((SP)) <-- (RD);
(SP) <-- (SP) - 2;   [Empilha RC]
((SP)) <-- (RC);
(SP) <-- (SP) - 2;   [Empilha RB]
((SP)) <-- (RB);
(SP) <-- (SP) - 2;   [Empilha RA]
((SP)) <-- (RA);
```

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Salva na pilha de programa uma cópia dos registros de uso geral da UCP/ASTROP.

CRG

Restaura registro de uso geral

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

Operação:
```
(RA) <-- ((SP));   [Desempilha RA]
(SP) <-- (SP) + 2;
(RB) <-- ((SP));   [Desempilha RB]
(SP) <-- (SP) + 2;
(RC) <-- ((SP));   [Desempilha RC]
(SP) <-- (SP) + 2;
(RD) <-- ((SP));   [Desempilha RD]
(SP) <-- (SP) + 2;
(RE) <-- ((SP));   [Desempilha RE]
(SP) <-- (SP) + 2;
(RF) <-- ((SP));   [Desempilha RF]
(SP) <-- (SP) - 2.
```

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Restaura da pilha de programa os conteúdos registros de uso geral da UCP/ASTROP, lá salvos pela instrução SRG.

TRAP(n)

Desvio para "trap" n

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	0	0	0	1	1	n	n	n

Operação:
```
(SPS) <-- (SPS) - 2;  [Empilha PC atual]
((SPS)) <-- (PC);
(SPS) <-- (SPS) - 2;  [Empilha PSW atual]
((SPS)) <-- (PSW);
(PSW) <-- ("FFA0"H + 4 X nnn);
(PC) <-- ("FFA2"H + 4 X nnn);
```

Códigos de condição: C, V, Z, S, T1, T2, T3 recebem um novo contexto.

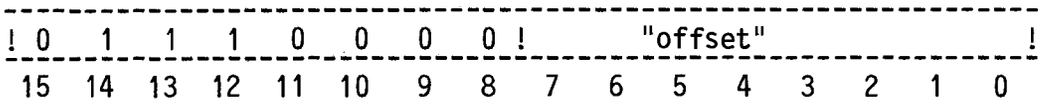
Descrição: Salva na pilha o PC e a PSW, e obtêm o vetor de "trap" nos endereços "FFA0"H + 4 X nnn e "FFA2"H + 4 X nnn da memória.

4.3.4 - INSTRUÇÕES COM PARÂMETRO

As instruções com parâmetro encontram-se descritas nesta seção na seguinte ordem:

DR

Desvio relativo incondicional



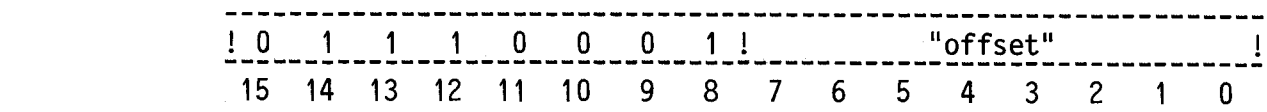
Operação: (PC) <-- (PC) + (2 X "offset").

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Provê uma maneira de transferir o controle do programa dentro de uma faixa de -128 a +127 palavras com uma instrução de uma única palavra. É importante notar que no cálculo do novo conteúdo do PC entra o valor do PC já incrementado de dois pelo procedimento de "fetch".

DCO

Desvio relativo se bit C for "1"



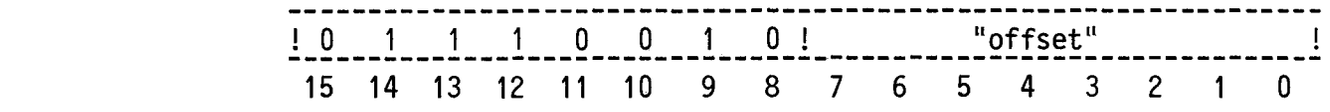
Operação: (PC)<-- (PC) + (2 X "offset") se C = 1.

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Testa o estado do código de condição C e efetua um desvio relativo se C igual a "1".

DCZ

Desvio relativo se bit C for "0"



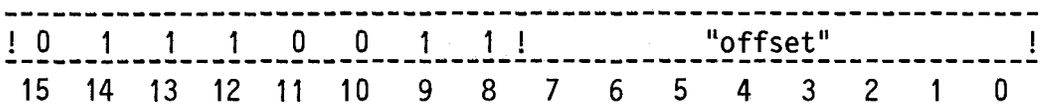
Operação: (PC) <-- (PC) + (2 X "offset") se C = 0.

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Testa o estado do código de condição C e efetua um desvio relativo se C for igual a "0".

DVO

Desvio relativo se bit V for "1"



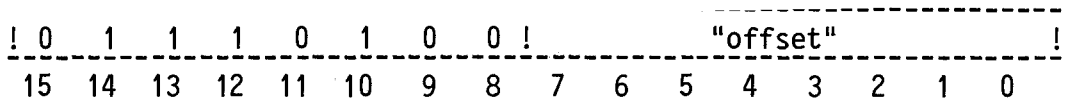
Operação: (PC)<-- (PC) + (2 X "offset") se V = 1.

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Testa o estado do código de condição V e efetua um desvio relativo se V for igual a "1".

DVZ

Desvio relativo se bit V for "0"



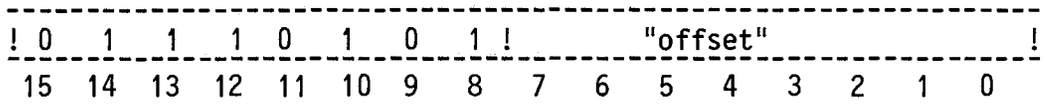
Operação: (PC)<-- (PC) + (2 X "offset") se V = 0.

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Testa o estado do código de condição V e efetua um desvio relativo se V for igual a "0".

DZO

Desvio relativo se bit Z for "1"



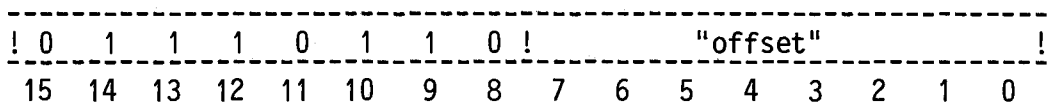
Operação: (PC) <-- (PC) + (2 X "offset") se Z = 1.

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Testa o estado do código de condição Z e efetua um desvio relativo se Z for igual a "1".

DZZ

Desvio relativo se bit for "0"



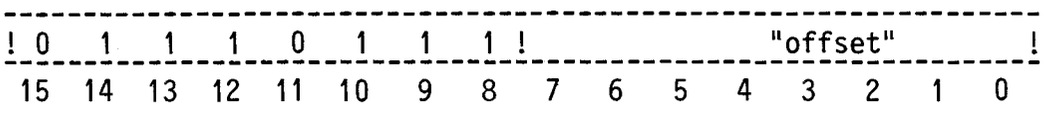
Operação: (PC) <-- (PC) + (2 X "offset") se Z = 0.

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Testa o estado do código de condição Z e efetua um desvio relativo se Z for igual a "0".

DSO

Desvio relativo se bit S for "1".



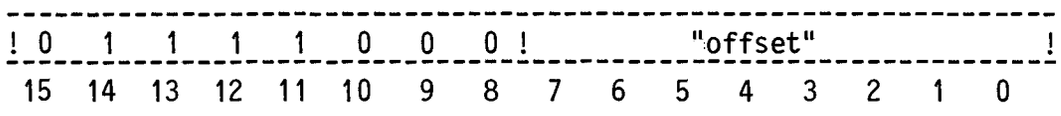
Operação: (PC) <-- (PC) + (2 X "offset") se S = 1.

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Testa o estado do código de condição S e efetua um desvio relativo se S for igual a "1".

DSZ

Desvio relativo se bit S for "0"



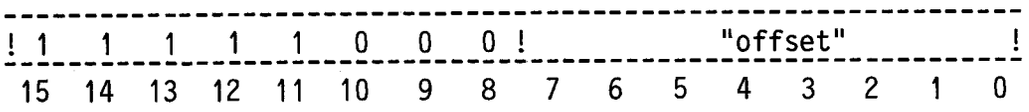
Operação: (PC) <-- (PC) + (2 X "offset") se S = 0.

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Testa o estado do código de condição S e efetua um desvio relativo se S for igual a "0".

DTAO

Desvio relativo se bit T1 for "1"



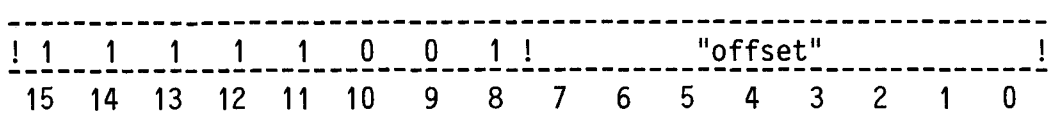
Operação: (PC) <-- (PC) + (2 X "offset") se T1 = 1.

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Testa o estado do código de condição T1 e efetua um desvio relativo se T1 for igual a "1".

DTAZ

Desvio relativo se bit T1 for "0"



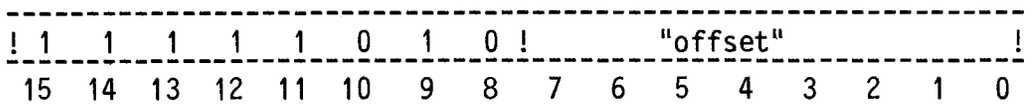
Operação: (PC) <-- (PC) + (2 X "offset") se T1 = 0.

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Testa o estado do código de condição T1 e efetua um desvio relativo se T1 igual a "0".

DTBO

Desvio relativo se bit T2 for "1"



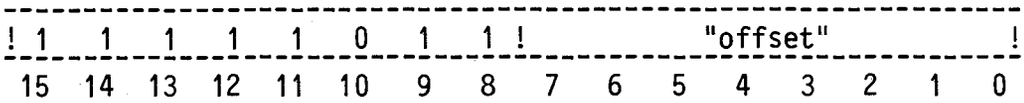
Operação: (PC) <-- (PC) + (2 X "offset") se T2 = 1.

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Testa o estado do código de condição T2 e efetua um desvio relativo se T2 igual a "1".

DTBZ

Desvio relativo se bit T2 for "0"



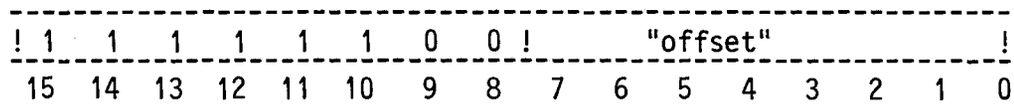
Operação: (PC) <-- (PC) + (2 X "offset") se T2 = 0.

Códigos de condição: C, V, Z, S, T1, T2, T3. não são afetados.

Descrição: Testa o estado do código de condição T2 e efetua desvio relativo se T2 igual a "0".

DTCO

Desvio relativo se bit T3 for "1"

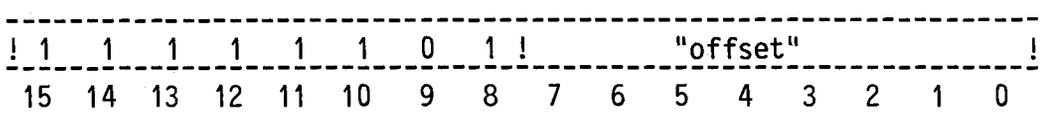


Operação: (PC) <-- (PC) + (2 X "offset") se T3 = 1.

Cõdigos de condição: C, V, Z, S, T3, T3, T3: não são afetados.

Descrição: Testa o estado do cõdigo de condição T3 e efetua um desvio relativo se T3 igual a "1".

Desvio relativo se bit T3 for "0"



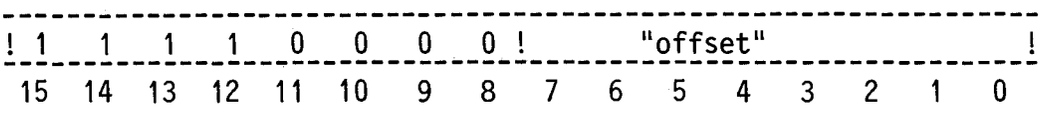
Operação: (PC) <-- (PC) + (2 X "offset") se T3 = 0.

Códigos de condição: C, V, Z, S, T3, T3, T3: não são afetados.

Descrição: Testa o estado do código de condição T3 e efetua um desvio relativo se T3 for igual a "0".

DGE

Desvio relativo se maior ou igual (números com sinal)



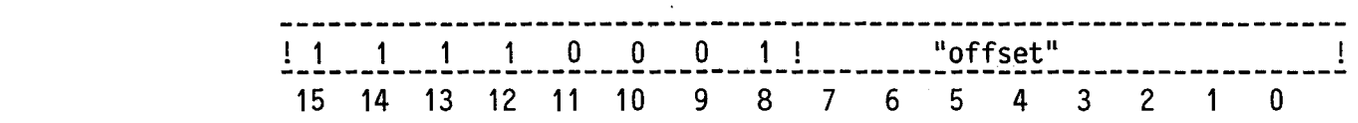
Operação: (PC) <-- (PC) + (2 X "offset") se S [*] V = 0.

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Causa um desvio relativo se os códigos de condição S e V são ambos iguais a "0" ou ambos iguais a "1"; o que ocorre se for subtraído de um número em complemento de dois um valor menor ou igual a ele.

DLT

Desvio relativo se menor (números com sinal)



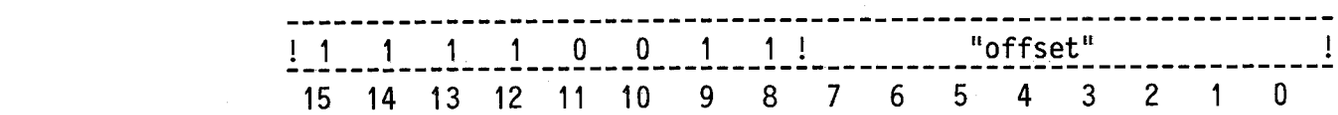
Operação: (PC) <-- (PC) + (2 X "offset") se S [*] V = 1.

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Causa um desvio relativo se os códigos de condição S e V apresentam estados opostos; o que ocorre se for subtraído de um número em complemento de dois um valor maior do que ele.

DLE

Desvio relativo se menor ou igual (números com sinal)



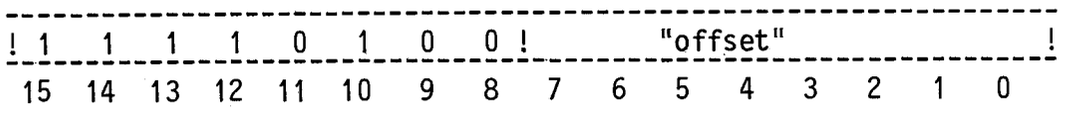
Operação: (PC) <-- (PC) + (2 X "offset") se Z [+] (S[*]Z) = 1.

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Causa um desvio relativo se os códigos de condição S e V apresentam estados opostos ou Z igual a "1"; o que ocorre se for subtraído de um número em complemento de dois um valor maior ou igual a ele.

DHI

Desvio relativo se maior (números sem sinal)

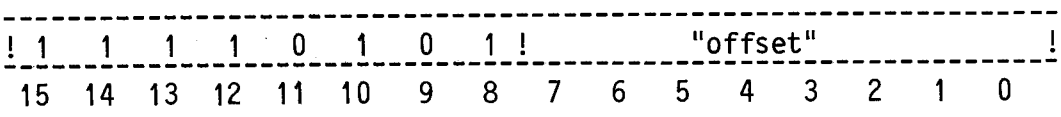


Operação: (PC) <-- (PC) + (2 X "offset") se C [+] Z = 0.

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Causa um desvio relativo se os códigos de condição C e Z são ambos iguais a "0"; o que ocorre se for subtraído de um número binário um valor binário menor do que ele.

DLS



Operação: (PC) <-- (PC) + (2 X "offset") se C[+] Z = 1.

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Causa um desvio relativo se um dos códigos de condição C e Z, ou ambos, é igual a "1"; o que ocorre se for subtraído de um número binário um valor binário maior ou igual a ele.

DHS

Desvio relativo se maior ou igual (números sem sinal)

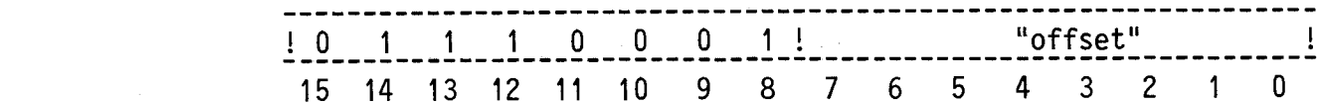
Operação: (PC) <-- (PC) + (2 X "offset") se C = 0.

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Causa um desvio relativo se o código de condição C for "0"; o que ocorre se for subtraído de um número binário um valor binário menor ou igual a ele. Esta instrução é análoga à instrução DCZ.

DLO

Desvio relativo se menor (números sem sinal)

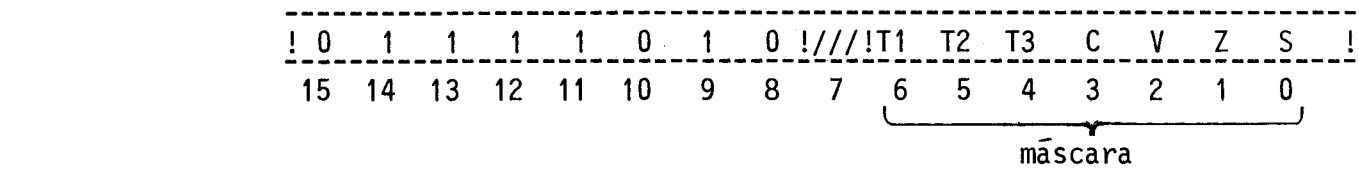


Operação: (PC) <-- (PC) + (2 X "offset") se C = 1.

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Causa um desvio relativo se o código de condição C for "1"; o que ocorre se for subtraído de um número binário um valor binário maior do que ele. Esta instrução é análoga à instrução DCO.

ERC



Operação: f6 = (T1[.](bit 6 da instru.))[*](bit 6 da instr.),
f5 = (T2[.](bit 5 da instru.))[*](bit 5 da instr.),
f4 = (T3[.](bit 4 da instru.))[*](bit 4 da instr.),
f3 = (C[.](bit 3 da instr.))[*](bit 3 da instr.),
f2 = (V[.](bit 2 da instr.))[+](bit 2 da instr.),
f1 = (Z[.](bit 1 da instr.))[*](bit 1 da instr.),
f0 = (S[.](bit 0 da instr.))[*](bit 0 da instr.),

f = f0 [+] f1 [+] f2 [+] f3 [+] f4 [+] f5 [+] f6.

Se f = 0, então faz:

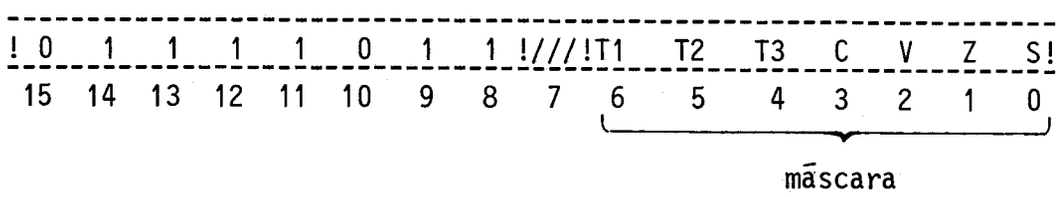
T1 <-- T1 [.] [-](bit 6 da instrução),
T2 <-- T2 [.] [-](bit 5 da instrução),
T3 <-- T3 [.] [-](bit 4 da instrução),
C <-- C [.] [-](bit 3 da instrução),
V <-- V [.] [-](bit 2 da instrução),
Z <-- Z [.] [-](bit 1 da instrução),
S <-- S [.] [-](bit 0 da instrução),
(PC) <-- (PC) + 2.

Caso contrário, não faz nada.

Descrição: Se todos os códigos de condição, que têm bits correspondentes na máscara (que acompanha a instrução) iguais a "1", estiverem iguais a "1", então faz estes códigos de condição iguais a "0" e incrementa o registro PC de dois (pula a próxima instrução que deve ter 16 bits). Caso contrário, não faz nada.

CF0

Faz códigos de condição iguais a "1"

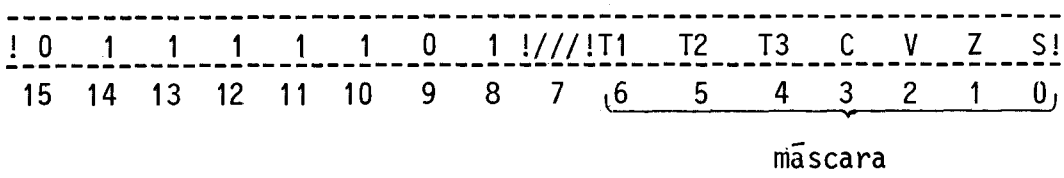


Operação: T1 <-- T1 [+] (bit 6 da instrução),
T2 <-- T2 [+] (bit 5 da instrução),
T3 <-- T3 [+] (bit 4 da instrução),
C <-- C [+] (bit 3 da instrução),
V <-- V [+] (bit 2 da instrução),
Z <-- Z [+] (bit 1 da instrução),
S <-- S [+] (bit 0 da instrução).

Descrição: São feitos iguais a "1" todos os códigos de condição que possuem os bits correspondentes na máscara (que acompanha a instrução) iguais a "1". Os outros códigos de condição não são afetados.

CFZ

Faz códigos de condição iguais a "0"

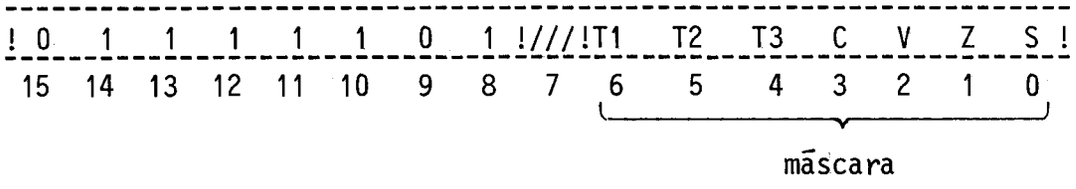


Operação: T1 <-- T1 [.] (bit 6 da instrução),
T2 <-- T2 [.] (bit 5 da instrução),
T3 <-- T3 [.] (bit 4 da instrução),
C <-- C [.] (bit 3 da instrução),
V <-- V [.] (bit 2 da instrução),
Z <-- Z [.] (bit 1 da instrução),
S <-- S [.] (bit 0 da instrução).

Descrição: São feitos iguais a "0" todos os códigos que possuem os bits correspondentes na máscara (que acompanha a instrução) iguais a "0". Os outros códigos de condição não são afetados.

CCC

Carrega códigos de condição

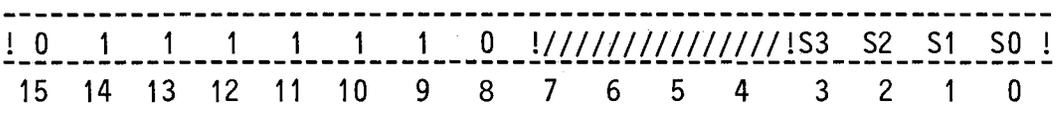


Operação: T1<-- (bit 6 da instrução),
T2<-- (bit 5 da instrução),
T3<-- (bit 4 da instrução),
C<-- (bit 3 da instrução),
V<-- (bit 2 da instrução),
Z<-- (bit 1 da instruçaõ),
S<-- (bit 0 da instrução).

Descrição: Armazena nos códigos de condição os bits correspondentes na máscara que acompanha a instrução.

CNM

Carrega nível mínimo de interrupção



Operação: Bits 11 a 8 da PSW<-- bits 3 a 0 da instrução.

Código de condiçaõ: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Armazena no campo da palavra de "status" do processador (PSW), referente ao nível mínimo de interrupção, os quatro bits S3 a S0. A partir daí só serão aceitas interrupções de nível maior ou igual àquele armazenado na (PSW). Observe que S3 a S0 igual a "0"H corresponde a habilitar interrupção de todos os níveis não-mascarados, e que S3 a S0 igual a "F"H equivale a desabilitar todas as interrupções, exceto a de nível 15 (de maior prioridade e que não pode nunca ser mascarada).

4.3.5 - INSTRUÇÕES PARA A UNIDADE ARITMÉTICA ASTROM

A unidade aritmética ASTROM opera com números em ponto fixo (inteiros) e com números em ponto flutuante (fracionários). Ela possui quatro registros de 32 bits acessíveis pelo usuário por programação, os quais são codificados como mostra a Tabela 4.1.

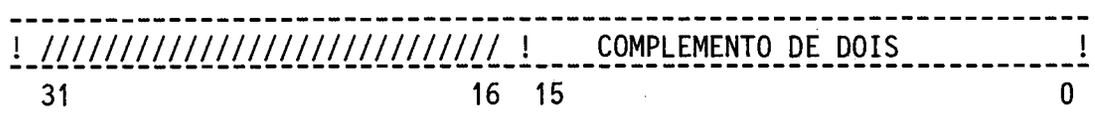
TABELA 4.1

CÓDIGO DOS REGISTROS INTERNOS DO ASTROM

CÓDIGO	REGISTRO AC1/AC2
00	R0
01	R1
10	R2
11	R3

A representação dos números no ASTROM é a da Figura 4.1. Os números em ponto fixo utilizam 16 bits representados em complemento de dois e variam de -32768 a +32767, conforme a Tabela 4.2. Os 16 bits mais significativos dos registros são zerados quando se opera com números em ponto fixo.

a) números inteiros



b) números em ponto flutuante

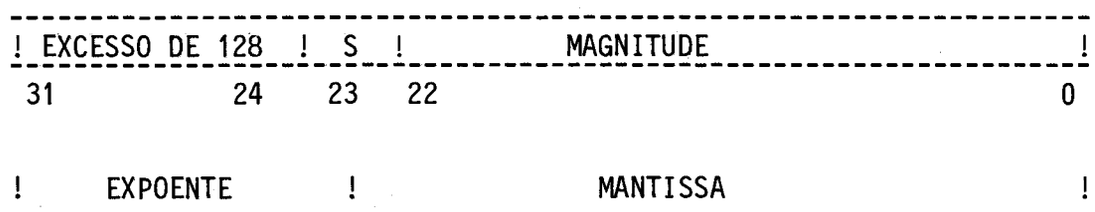


Fig. 4.1 - Representação dos números no ASTROM. a) e b).

Os números em ponto flutuante utilizam 32 bits, sendo oito bits correspondentes ao expoente e 24 bits correspondentes à mantissa.

O expoente é representado em excesso de 128 e varia de -127 a +127 na base 2, o que corresponde a uma variação de -38 a +38, aproximadamente, na base 10. A Tabela 4.3 mostra a faixa de valores do expoente, cuja representação é obtida somando 128 ("80"H) ao valor em complemento de dois do número desejado. Assim, -5 em complemento de dois é "FB"H e em excesso de 128 é "EB"H. O valor "00"H é reservado para representar o número 0 x 2^0.

VALOR DOS NÚMEROS EM PONTO FIXO

VALOR HEXADECIMAL (COMPLEMENTO DE DOIS)	VALOR DECIMAL
"7FFF"H	+32767
.	.
.	.
.	.
"0001"H	+1
"0000"H	0
"FFFF"H	-1
.	.
.	.
.	.
"8000"H	-32768

TABELA 4.4

CÓDIGOS DE CONDIÇÃO DO ASTROM

CÓDIGO DE CONDIÇÃO	SIGNIFICADO
UPFOK	"1", circuitos do ASTROM energizados e no modo remoto de funcionamento; "0", caso contrário.
ZUPF	"1", resultado igual a zero; "0", caso contrário.
SUPF	"0", resultado positivo; "1", resultado negativo.
DIVZ	"1", divisão por zero; "0", caso contrário.
OVFL	"1", "overflow" do resultado; "0", caso contrário.
UNDF	"1", "underflow" do resultado; "0", caso contrário.
CVER	"1", erro de conversão flutuante para inteiro; "0", caso contrário.

A ativação dos códigos de condição DIVZ, OVFL, UNDF ou CVER, ou a desativação do código de condição UPFOK causam um "trap" da unidade aritmética ASTROM, na UCP/ASTROP (Seção 2.2.7).

As instruções para o ASTROM fazem parte do conjunto de instruções do ASTROP, num total de 32 instruções, cujas operações podem ser classificadas em:

1) Operações aritméticas que compreendem a adição, subtração, multiplicação e divisão, tanto em ponto fixo como em ponto flutuante.

2) Operações de lógica, constituídas pelas instruções de negação e valor absoluto em ponto fixo e flutuante e comparação em ponto flutuante.

3) Operações de conversão que consiste na mudança de representação de ponto fixo para flutuante e vice-versa.

4) Operações de transferência que compreendem as instruções de transferência de dados entre a unidade aritmética e o computador ASTROP.

A seguir serão descritas as instruções da unidade aritmética ASTROM, de acordo com as suas classes de operação e na seguinte ordem:

a) Instruções aritméticas:

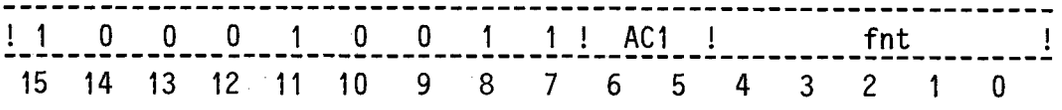
b) Instruções de lógica:

c) Instruções de conversão:

d) Instruções de transferência:

SMIP

Soma inteiro



Operação: (AC1) <-- (AC1) + (fnt).

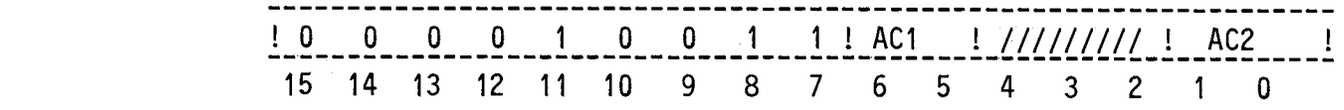
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Soma o número inteiro armazenado no registro AC1 do ASTROM com o operando fonte e guarda o resultado no registro AC1. O conteúdo original do endereço fonte não é alterado. O acesso ao operando fonte no ASTROP é de palavra.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

SMIM

Soma inteiro



Operação: (AC1) <-- (AC1) + (AC2).

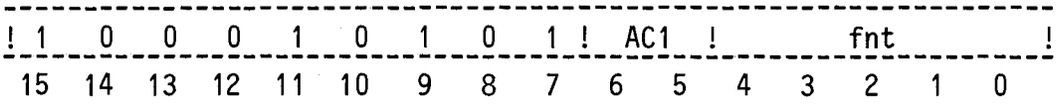
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Instrução análoga à SMIP, só que opera apenas com registros internos do ASTROM.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

SUIP

Subtrai inteiro



Operação: (AC1) <-- (AC1) - (fnt).

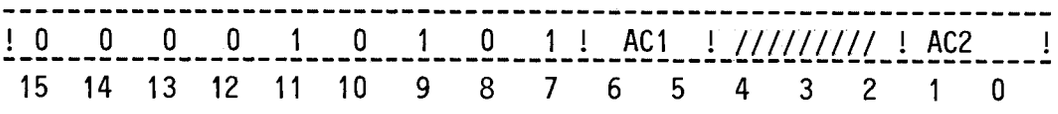
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Subtrai o número inteiro armazenado no endereço fonte do número inteiro armazenado no registro AC1 do ASTROM e guarda o resultado no registro AC1. O conteúdo original do endereço fonte não é alterado. O acesso ao operando fonte no ASTROP é de palavra.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

SUIM

Subtrai inteiro



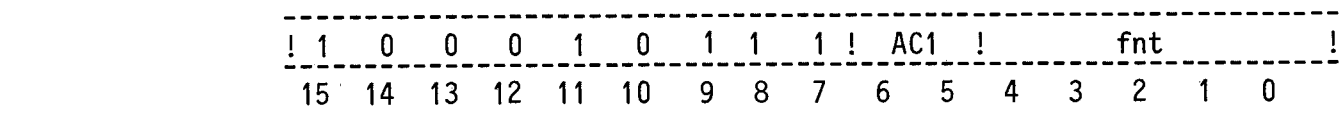
Operação: (AC1) <-- (AC1) - (AC2).

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Instrução análoga a SUIP, só que opera apenas com registros internos do ASTROM.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

MLIP



Operação: (AC1) <-- (AC1) x (fnt).

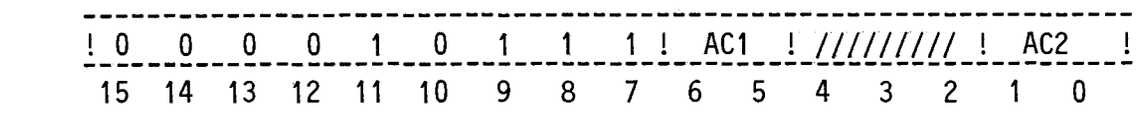
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Multiplica o número inteiro armazenado no endereço fonte com o número inteiro armazenado no registro AC1 do ASTROM e guarda o resultado no registro AC1. O conteúdo original do endereço fonte não é alterado. O acesso ao operando fonte no ASTROP é de palavra.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

MLIM

Multiplica inteiro



Operação: (AC1) <-- (AC1) x (AC2).

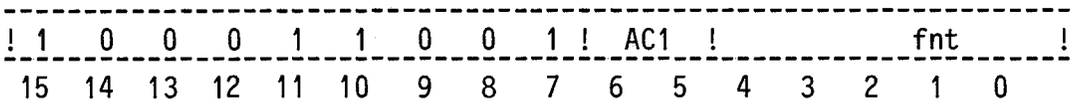
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Instrução análoga a MLIP, só que opera apenas com registros internos do ASTROM.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

DVIP

Divide inteiro



Operação: (AC1) <-- (AC1) / (fnt).

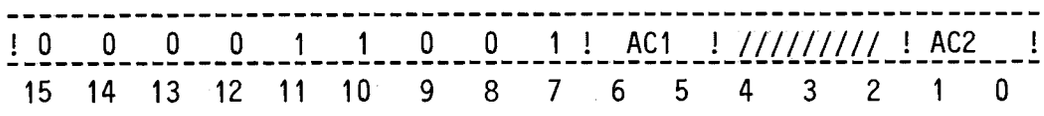
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Divide o número inteiro armazenado no registro AC1 do ASTROM pelo número inteiro armazenado no registro fonte do ASTROP e guarda o resultado no registro AC1. O conteúdo original do endereço fonte não é alterado. O acesso ao operando fonte no ASTROP é de palavra.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

DVIM

Divide inteiro



Operação: (AC1) <-- (AC1) / (AC2).

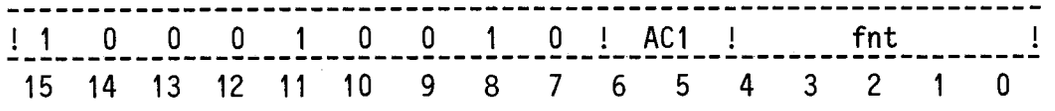
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Instrução análoga a DVIP, só que opera apenas com registros internos do ASTROM.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

SMPFP

Soma ponto flutuante



Operação: (AC1) <-- (AC1) + (fnt).

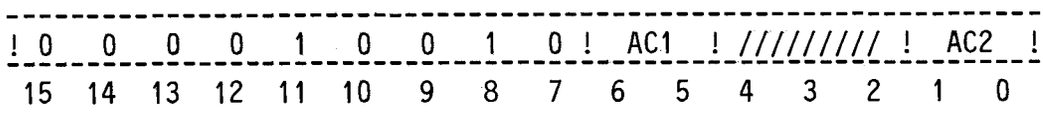
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Soma, em ponto flutuante, o operando armazenado no endereço fonte do ASTROP com o conteúdo do registro AC1 do ASTROM e guarda o resultado no registro AC1. O conteúdo original do endereço fonte não é alterado. O acesso ao operando fonte no ASTROP é de palavra longa.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

SMPFM

Soma ponto flutuante



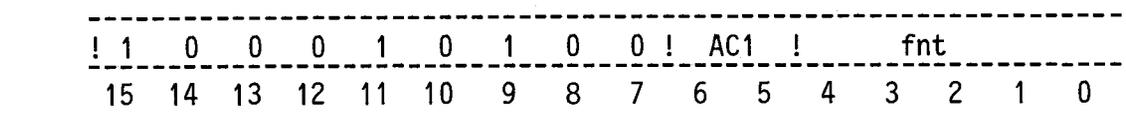
Operação: (AC1) <-- (AC1) + (AC2).

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Instrução análoga à SMPFP, só que opera apenas com registros internos do ASTROM.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

Subtrai ponto flutuante



Operação: (AC1) <-- (AC1) - (fnt).

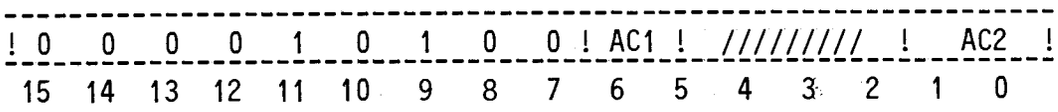
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Subtrai o número representado em ponto flutuante, armazenado no endereço fonte, do número também representado em ponto flutuante armazenado no registro AC1 do ASTROM. O resultado é guardado no registro AC1. O conteúdo original do endereço fonte não é alterado. O acesso ao operando fonte no ASTROP é de palavra longa.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

SUPFM

Subtrai ponto flutuante



Operação: (AC1) <-- (AC1) - (AC2).

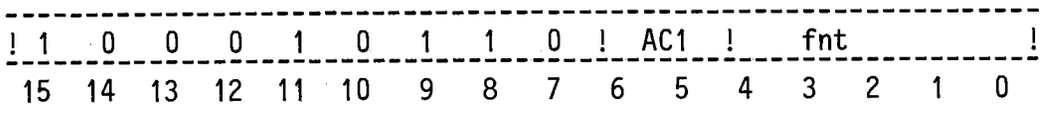
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Instrução análoga a SUPFP, só que opera apenas com registros internos do ASTROM.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

MLPFP

Multiplica ponto flutuante



Operação: (AC1) <-- (AC1) x (fnt).

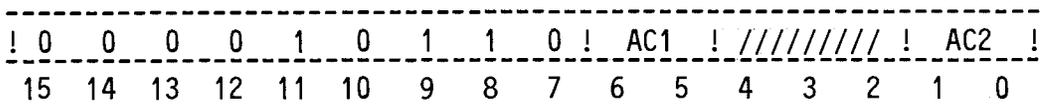
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Multiplica em ponto flutuante o operando armazenado no endereço fonte do ASTROP com o conteúdo do registro AC1 do ASTROM e guarda o resultado no registro AC1. O conteúdo original do endereço fonte não é alterado. O acesso ao operando fonte no ASTROP é de palavra longa.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

MLPFM

Multiplica ponto flutuante



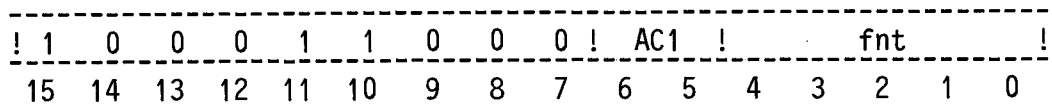
Operação: (AC1) <-- (AC1) x (AC2).

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Instrução análoga a MLPFP, só que opera apenas com registros internos do ASTROM.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

Divide ponto flutuante



Operação: (AC1) <-- (AC1) / (fnt).

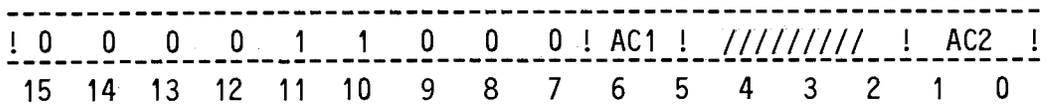
Códigos de condiçaõ: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Divide em ponto flutuante o operando armazenado no registro AC1 do ASTROM pelo operando armazenado no endereço fonte do ASTROP e guarda o resultado no registro AC1. O conteúdo original do endereço fonte no ASTROP não é alterado. O acesso ao operando fonte no ASTROP é de palavra longa.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

DVPFM

Divide ponto flutuante



Operação: (AC1) <-- (AC1) / (AC2).

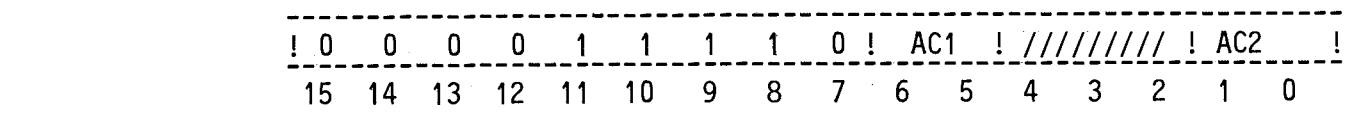
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Instrução análoga a DVPFP, só que opera apenas com registros internos do ASTROM.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

NGIN

Nega inteiro



Operação: (AC1) <-- -(AC2).

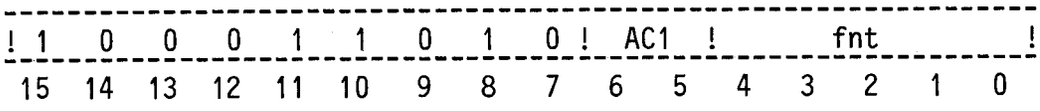
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Inverte o sinal do número inteiro armazenado no registro AC2 do ASTROM e guarda o resultado no registro AC1 do ASTROM. O conteúdo original registro AC2 não é alterado.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

NGPFP

Nega ponto flutuante



Operação: (AC1) <-- -(fnt).

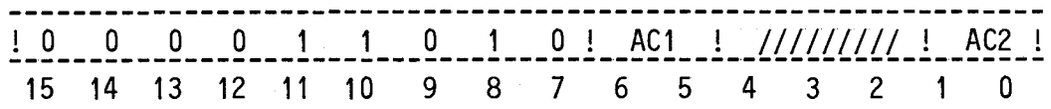
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Inverte o sinal do número representado em ponto flutuante armazenado no endereço fonte do ASTROP e guarda o resultado no registro AC1 do ASTROM. O conteúdo original do endereço fonte não é alterado. O acesso ao operando fonte no ASTROP é de palavra longa.

Código de condição do ASTROM: alterados de acordo com a Tabela 4.4.

NGPFM

Nega ponto flutuante



Operação: (AC1) <-- -(AC2).

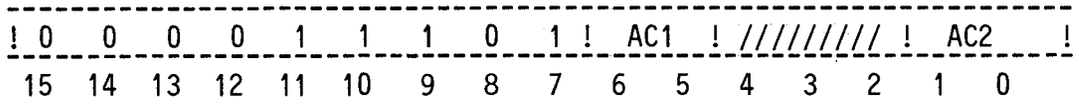
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Instrução análoga a NGPFP, só que opera apenas com registros internos do ASTROM.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

ABSI

Absuluto inteiro



Operação: (AC1) <-- módulo do (AC2).

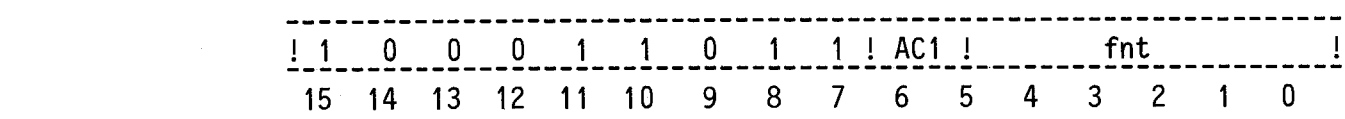
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Armazena no registro AC1 o valor positivo correspondende ao número inteiro armazenado no registro AC2. O conteúdo original do registro AC2 não é alterado.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

CMPFP

Compara ponto flutuante



Operação: (AC1) - (fnt).

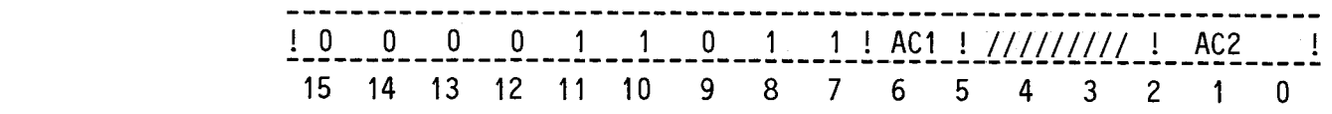
Códigos de condição : C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Subtrai o número representado em ponto flutuante armazenado no endereço fonte do número também representado em ponto flutuante armazenado no registro AC1 do ASTROM. Os conteúdos do endereço fonte e do registro AC1 não são alterados. O acesso ao operando fonte no ASTROP é de palavra longa.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

CMPFM

Compara ponto flutuante



Operação: (AC2) - (AC2).

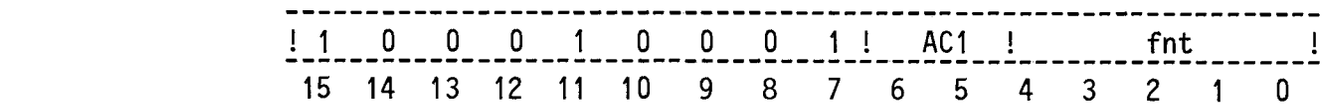
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Instrução análoga a CMPFP, só que opera apenas com registros internos do ASTROM.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

IPFP

Converte inteiro em ponto flutuante



Operação: (AC1) /ponto flutuante/ <-- (fnt) /inteiro/.

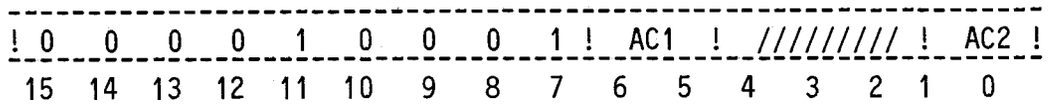
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Transforma o número inteiro armazenado no endereço fonte em um número representado em ponto flutuante e guarda o resultado no registro AC1 do ASTROM. O conteúdo original do endereço fonte não é alterado. O acesso ao operando fonte no ASTROP é de palavra.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

IPFM

Converte inteiro em ponto flutuante



Operação: (AC1) /ponto flutuante/ <-- (AC2) /inteiro/.

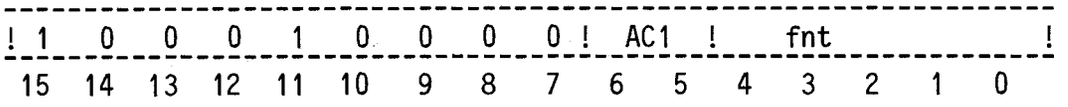
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Instrução análoga a IPFP, só que opera apenas com registros internos do ASTROM.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

PFIP

Converte ponto flutuante em inteiro



Operação: (AC1) /inteiro/ <-- (fnt) /ponto flutuante/.

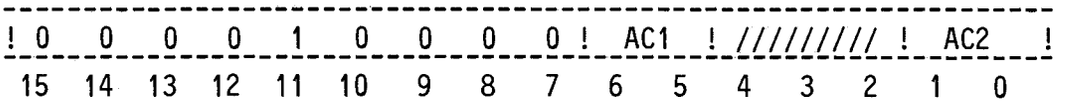
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Transforma o número representado em ponto flutuante armazenado no endereço fonte em um número inteiro e guarda o resultado no registro AC1 do ASTROM. O acesso ao operando fonte no ASTROP é de palavra longa.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

PFIM

Converte ponto flutuante em inteiro



Operação: (AC1) /ponto flutuante/ <-- (AC2) /inteiro/.

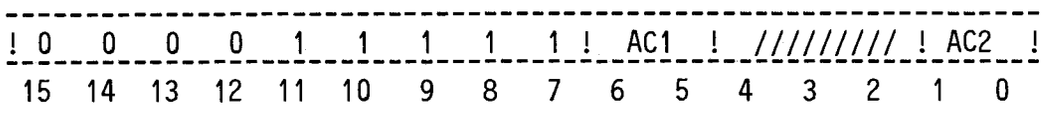
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Instrução análoga a PFIP, só que opera apenas com registros internos do ASTROM.

Códigos de condição do ASTROM: alterados de acordo com a Tabela 4.4.

CPRGI

Cópia registro interno



Operação: (AC1) <-- (AC2).

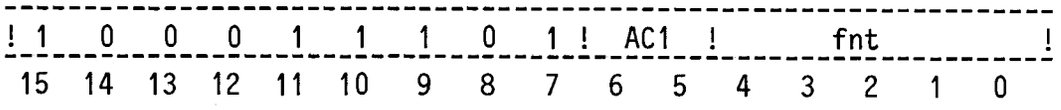
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Copia o conteúdo do registro AC2 do ASTROM no registro AC1 do ASTROM. O conteúdo do registro AC2 não é alterado.

Códigos de condição do ASTROM: todos recebem zero.

CRI

Carrega inteiro



Operação: (AC1) <-- (fnt).

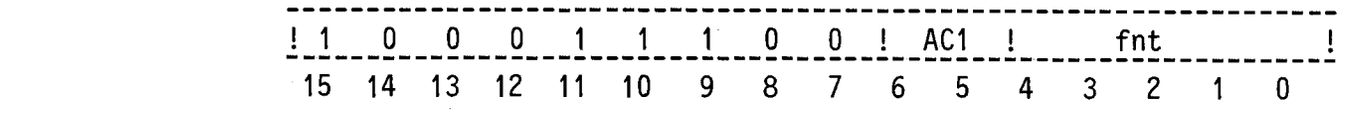
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Carrega registro AC1 do ASTROM com o número inteiro armazenado no endereço fonte do ASTROP. O conteúdo original do endereço fonte do ASTROP não é alterado. O acesso ao operando fonte é a palavra.

Códigos de condição do ASTROM: todos recebem zero.

CRPF

Carrega ponto flutuante



Operação: (AC1) <-- (fnt).

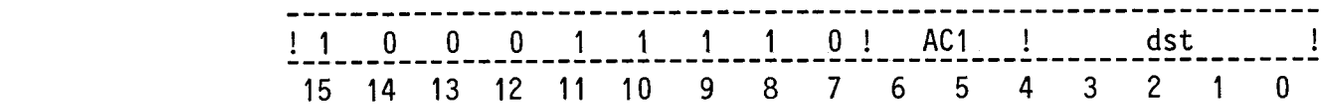
Códigos de condição: C, V, Z, S, T1, T2, T3: não são alterados.

Descrição: Carrega registro AC1 do ASTROM com o número representado em ponto flutuante armazenado no endereço fonte do ASTROP. O conteúdo original do endereço fonte no ASTROP não é alterado. O acesso ao operando fonte é de palavra longa.

Códigos de condição do ASTROM: todos recebem zero.

AZPF

Armazena ponto flutuante



Operação: (dst) <-- (AC1).

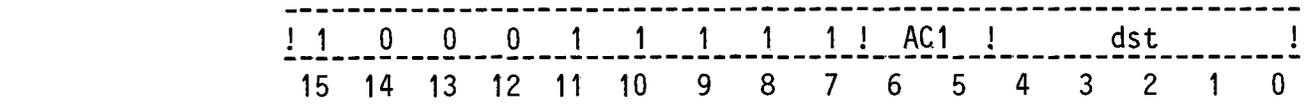
Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Armazena o número representado em ponto flutuante guardado no registro AC1 do ASTROM, no endereço destino do ASTROP. O conteúdo original do registro AC1 não é alterado. O acesso ao endereço destino no ASTROP é de palavra longa.

Códigos de condição do ASTROM: todos recebem zero.

AZI

Armazena inteiro



Operação: (dst) <-- (AC1).

Códigos de condição: C, V, Z, S, T1, T2, T3: não são afetados.

Descrição: Armazena o número inteiro, guardado no registro AC1 do ASTROM, no endereço destino do ASTROP. O conteúdo original do registro AC1 não é alterado. O acesso ao endereço destino no ASTROP é de palavra.

Códigos de condição do ASTROM: todos recebem zero.

CAPÍTULO 5

DETALHES DE PROGRAMAÇÃO

Neste capítulo são abordados alguns detalhes de programação que utilizam adequadamente o conjunto de instruções do computador ASTROP.

5.1 - SUB-ROTINAS

Sub-rotinas no ASTROP são chamadas com o uso da instrução JSB que referencia dois operandos:

JSB, operando 1, operando 2.

O operando 1 é salvo na pilha de programa e no seu lugar é colocado o conteúdo do registro PC. A seguir este registro recebe o operando 2 que designa o endereço inicial da sub-rotina em questão. Este mecanismo permite que a passagem de parâmetros para uma sub-rotina possa se dar de várias maneiras.

Em uma destas maneiras o parâmetro é o próprio operando 1 que, neste caso, pode ser acessado pela sub-rotina com o uso do modo de endereçamento topo da pilha. Ressalte-se que um eventual parâmetro de retorno (resultado da sub-rotina) pode ser deixado no topo da pilha para, ao final, assumir o lugar do parâmetro passado inicialmente para a sub-rotina.

Uma outra maneira (Figura 5.1) é os parâmetros estarem localizados no corpo do programa principal. Neste caso, o operando 1 pode estar em um registro de uso geral, por exemplo. Assim, no início da execução da sub-rotina, este registro de uso geral armazenará o endereço onde estão os parâmetros da sub-rotina, que podem ser acessados com o uso do modo de endereçamento registro indireto. No final da sub-rotina deve-se alterar o conteúdo do registro de uso geral em questão, de

forma que o retorno da sub-rotina se dê para o corpo do programa principal no endereço imediatamente após os parâmetros. Ressalte-se que o conteúdo original do registro de uso geral utilizado não é perdido com a execução da sub-rotina.

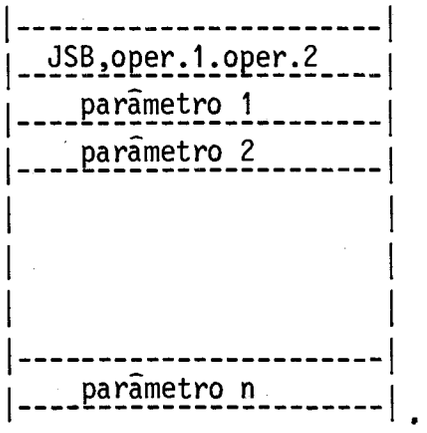


Fig. 5.1 - JSB: exemplo de passagem de parâmetros.

A passagem de resultados booleanos de sub-rotinas pode ser feita com a utilização dos códigos de condição T1, T2 e T3 que não são utilizados de forma automática pela UCP/ASTROP, como os códigos de condição C, V, Z, e S.

5.2 - INTERRUPÇÕES

O mecanismo de atendimento de interrupções causa o armazenamento de um novo conteúdo para a palavra de "status" do processador (PSW). Assim, o nível mínimo de interrupção (S3 a S0), que faz parte da nova PSW, pode ser qualquer e não tem nenhum relacionamento direto com o nível da interrupção atendida. Isto é, o atendimento de uma interrupção de nível i não acarreta automaticamente o não reconhecimento das interrupções de níveis 0 a i.

A instrução CNM pode ser usada para desabilitar o atendimento das interrupções dos níveis 0 a 14. E as instruções CMK, SMK, MKI e DMI alteram a máscara de interrupções (registro MK).

5.3 - CÓDIGO OBJETO INDEPENDENTE DA POSIÇÃO

Além do uso da base do programa (B3 a B0) para realocar código objeto na memória do ASTROP, o emprego dos modos de endereçamento imediato, relativo e relativo indireto torna o código objeto independente da posição onde ele venha a ser armazenado na memória do ASTROP para ser executado.

5.4 - EXECUÇÃO DE PROGRAMAS PASSO A PASSO

O uso do bit de "trap" de "breakpoint" (bit B) da palavra de "status" do processador implementa uma maneira eficiente de realizar "breakpoints" e de executar programas passo a passo.

Sempre que esse bit é feito igual a "1", uma instrução é executada completamente antes do "trap" de "breakpoint" ocorrer. A alteração do bit de "trap" de "breakpoint" apenas pode ser feita com o armazenamento de um novo conteúdo na PSW. Assim, a rotina de atendimento de "trap" de "breakpoint" ao executar a instrução RTT pode fazer o bit B igual a "1", o que causará a execução de uma instrução e, a seguir, o retorno para a rotina de "trap" de "breakpoint". Evidentemente o vetor de "trap" de "breakpoint" não pode ter o bit B igual a "1".

APÊNDICE A

TEMPO DE EXECUÇÃO DAS INSTRUÇÕES

O tempo de execução de uma instrução depende da própria instrução e dos modos de endereçamento utilizados. No caso mais geral ele é o somatório de sete parcelas, a saber:

a) Tempo básico: formado pelos ciclos da UCP/ASTROP não inclusos nas parcelas seguintes (por exemplo no "fetch" da instrução).

b) tempo de busca do operando fonte,

c) tempo de busca do operando destino,

d) tempo de guarda do operando fonte,

e) tempo de guarda do operando destino,

f) tempo de acesso à pilha de programa ou do sistema, não incluído nas parcelas anteriores,

g) tempo de acesso aos vetores de "trap".

Neste apêndice os tempos são expressos em múltiplos de dois tipos de ciclos de máquina da UCP/ASTROP:

1) Ciclo de máquina onde não ocorre entrada ou saída (E/S) de dados na UCP/ASTROP através do barramento do sistema (BASIS). Este tipo de ciclo é denotado por T.

2) Ciclo de máquina, denotado por T*, onde ocorre E/S de dados na UCP/ASTROP através do BASIS.

No protótipo da UCP/ASTROP T é igual a 250 nanossegundos, enquanto T* é a soma de uma parcela fixa de 400 nanossegundos com o tempo de resposta (tRD) do dispositivo de E/S acessado através do BASIS. Ou seja:

T = 250 ns.

T* = 400 ns + tRD.

O tempo tRD é a medida do atraso entre as bordas de descida dos sinais REQOP* e PEROP* do BASIS. Na Tabela A.1 encontram-se alguns valores típicos para tRD no protótipo do computador ASTROP.

TABELA A.1

VALORES TÍPICOS PARA tRD NO PROTÓTIPO DO COMPUTADOR ASTROP

DISPOSITIVO ACESSADO NO BASIS	tRD (TÍPICO)
Memória RAM	350 ns
Vetores de "Reset", "Traps" e interrupções	500 ns
Temporizador programável	500 ns
Interface de comunicação serial (ICS)	500 ns

Os tempos de busca e guarda de operando fonte ou destino na UCP/ASTROP, além de dependerem dos modos de endereçamento utilizados, podem variar de uma instrução para outra. Assim sendo, na Tabela A.2 encontram-se listados os tempos gastos na busca e guarda de operandos em três colunas distintas (colunas A, B e C). E, nas Tabelas A.3 a A.8 estão referenciadas quais colunas da Tabela A.2 devem ser utilizadas no cômputo dos tempos de execução das instruções.

Nas Tabelas A3 a A.7 encontram-se, respectivamente, os tempos de execução das instruções com dois operandos, com um operando, com zero operandos, com parâmetro e para a unidade aritmética ASTROM. Finalmente, na Tabela A.8, estão os tempos de resposta da UCP/ASTROP a alguns eventos, como atendimento de "traps" e interrupções.

Como exemplo de cálculo de tempo de execução, suponha-se a instrução:

ADD (IM.AB ("00FB"H, "CØØØ"H)) .

Das Tabelas A.2 a A.3 obtêm-se:

$$\begin{aligned}
\text{tempo de execução} &= 2T + 1T^*\,(RAM) + \\
&+ 2T + 1T^*\,(RAM) \begin{bmatrix}\text{coluna A}\\ \text{imediato}\end{bmatrix} + \\
&+ 3T + 2T^*\,(RAM) \begin{bmatrix}\text{coluna A}\\ \text{absoluto}\end{bmatrix} + \\
&+ 1T + 1T^*\,(RAM) \begin{bmatrix}\text{coluna B}\\ \text{absoluto}\end{bmatrix} = \\
&= 8T + 5T^*\,(RAM) = \\
&= 8 \times 0{,}25 + 5\times(0{,}4+0{,}35)\,\mu s = \\
&= 5{,}75\ \mu s.
\end{aligned}$$

Como outro exemplo, tome-se a instrução:

MOVB (IM.AB("00FA"H, "FFØ2"H))

que programa um "port" na interface de comunicação serial (ICS).

Das Tabelas A.2 e A.3 obtém-se:

$$\text{tempo de execução} = 2T + 1T^*(\text{RAM}) +$$

$$+ 2T + 1T^*(\text{RAM}) \begin{bmatrix}\text{coluna A}\\ \text{imediato}\end{bmatrix} +$$

$$+ 4T + 2T^*(\text{RAM+ICS}) \begin{bmatrix}\text{coluna C}\\ \text{absoluto}\end{bmatrix} =$$

$$= 8T + 3T^*(\text{RAM}) + 1T^*(\text{ICS}) =$$

$$= 8 \times 0{,}25 + 3 \times (0{,}4+0{,}35) + (0{,}4+0{,}5) =$$

$$= 5{,}15\ \mu s.$$

TABELA A.2

TEMPOS DE BUSCA E GUARDA OPERANDO

MODO DE ENDEREÇAMENTO	TEMPOS DE EXECUÇÃO		
	COLUNA A	COLUNA B	COLUNA C
Registro direto	-	-	-
Registro indireto	2T + 1T*	1T + 1T*	3T + 1T*
Registro indexado	3T + 2T*	1T + 1T*	4T + 2T*
Registro indexado indireto	4T + 3T*	1T + 1T*	5T + 3T*
Ponteiro da pilha	2T	5T	6T
Ponteiro autodecrementado	2T + 1T*	1T + 1T*	3T + 1T*
Ponteiro autoincrementado	2T + 1T*	1T + 1T*	3T + 1T*
Topo da pilha	2T + 1T*	1T + 1T*	3T + 1T*
Imediato	2T + 1T*	1T + 1T*	3T + 1T*
Absoluto	3T + 2T*	1T + 1T*	4T + 2T*
Relativo	3T + 2T*	1T + 1T*	4T + 2T*
Relativo indireto	4T + 3T*	1T + 1T*	5T + 3T*

Observação 1 : Tempos expressos em ciclos de máquina sem E/S (T) e com E/S (T*).

TABELA A.3

TEMPOS DE EXECUÇÃO DAS INSTRUÇÕES COM DOIS OPERANDOS

INSTRUÇÃO	TEMPOS DE EXECUÇÃO					
	BÁSICO	BUSCA OPERANDO		GUARDA OPERANDO		ACESSO À PILHA
		FONTE	DESTINO	FONTE	DESTINO	
CMP (B)	2T + 1T*	Coluna A	Coluna A	-	-	-
COC (B), CZC (B) (Observação 3)	2T + 1T*	Coluna A	Coluna A	-	-	-
ADD (B), DAC (B), SUB (B), DSB (B) AND (B), OR (B), XOR (B), XNR (B)	2T + 1T*	Coluna A	Coluna A	-	Coluna B	-
MSK (B) (Observação)4)	2T + 1T*	Coluna A	Coluna A	-	Coluna B	-
EXC (B)	3T + 1T*	Coluna A	Coluna A	Coluna B	Coluna B	-
DJZ	6T + 1T* (Observação 5)	Coluna A	Coluna A	Coluna B	-	-
JSB (Observação 6)	6T + 1T*	Coluna A	Coluna A	Coluna B	-	1T*
MOV (B)	2T + 1T*	Coluna A	-	-	Coluna C	-

Observações: 1 - Tempos expressos em ciclos de máquina sem E/S (T) e com E/S (T*).

2 - As colunas referenciadas são colunas da Tabela A.2.

3 - No caso de os modos de endereçamento serem ambos registro direto, adicionar 1T.

4 - No caso de o modo de endereçamento do operando fonte ser registro direto e do modo de endereçamento do operando destino não ser registro direto, adicionar 1T.

5 - Se não ocorrer desvio, subtrair 2T.

6 - No caso de um dos modos de endereçamento ser registro direto, subtrair 2T.

TABELA A.4

TEMPOS DE EXECUÇÃO DAS INSTRUÇÕES COM UM OPERANDO

INSTRUÇÃO	TEMPOS DE EXECUÇÃO			
	BÁSICO	BUSCA OPERANDO	GUARDA OPERANDO	ACESSO À PILHA
NEG (B), INC (B), DCR (B), CMT (B), ADC (B), SB (B)	2T + 1T*	Coluna A	Coluna B	-
ICT (B), DCT (B), SWB	3T + 1T*	Coluna A	Coluna B	-
ABS (B)	3T + 1T*	Coluna A	Coluna B (Observação 3)	-
SLD (B), SDC (B), RDC (B), SAD (B), RTD (B), SLE (B), SEC (B), REC (B), RTE (B)	4T + 1T* (Observação 4)	Coluna A	Coluna B	-
SLDL, SDCL, RDCL, SADL, RTDL, SLEL, SECL, RECL, RTEL	3T + 1T*	2T	1T	-
RET	5T + 1T* (Observação 5)	Coluna A	Coluna B	1T*
STZ (B), STO (B), SMK, SSPS	2T + 1T*	-	Coluna C	-
SLP	3T + 1T*	-	Coluna C	-
SXT	3T + 1T*	-	Coluna C	-
SFGMB	3T + 1T* (Observação 4)	-	Coluna C	-
CPZ, CLP, CMK	2T + 1T*	Coluna A	-	-
JMP, CSPS	3T + 1T*	Coluna A	-	-
DMI	4T + 1T*	Coluna A	-	-
MKI	4T + 1T* (Observação 5)	Coluna A	-	-

Observações: 1 - Tempos expressos em ciclos de máquina sem E/S (T) e com E/S (T*).

2 - As colunas referenciadas são colunas da Tabela A.2.

3 - Não adicionar esta parcela se o operando originalmente for positivo.

4 - Se o modo de endereçamento não é registro direto, subtrair 1T.

5 - Se o modo de endereçamento é registro direto, subtrair 1T.

TABELA A.5

TEMPOS DE EXECUÇÃO DAS INSTRUÇÕES COM ZERO OPERANDOS

INSTRUÇÃO	TEMPOS DE EXECUÇÃO		
	BÁSICO	ACESSO À PILHA	ACESSO AOS VETORES
HALT	3T + 1T*	-	-
WAIT	4T + 1T*	-	-
NOP	2T + 1T*	-	-
RST (Observação 2)	3T + 1T*	-	-
EXS	18T + 1T*	4T*	-
RTT/RTI	9T + 1T*	2T*	-
BTP	16T + 1T*	2T*	2T*
SRG	9T + 1T*	6T*	-
CRG	15T + 1T*	6T*	-
TRAPn	14T + 1T*	2T*	2T*

Observações: 1 - Tempos expressos em ciclos de máquina sem E/S (T) e com E/S (T*).

2 - Computadas apenas as microoperações que disparam a ativação do circuito de "reset" do computador ASTROP.

TABELA A.7

TEMPOS DE EXECUÇÃO DAS INSTRUÇÕES PARA O ASTROM

INSTRUÇÃO	TEMPOS DE EXECUÇÃO		
	BÁSICO	BUSCA OPERANDO	GUARDA OPERANDO
PFIM, IPFM, SMPFM, SMIM, SUPFM, SUIM, MLPFM, MLIM, DVPFM, DVIM, NGPFM, CMPFM, ABPF, ABSI, NGIN, CPRGI	7T + 1T*	-	-
IPFP, SMIP, SUIP, MLIP, DVIP, CRI	12T + 1T*	Coluna A	-
PFIP, SMPFP, SUPFP, MLPFP, DVPFP, NGPFP, CMPFP, CRPF	17T + 1T*	2T	-
AZI	8T + 1T*	-	Coluna C
AZPF	14T + 1T*	-	2T

Observações: 1 - Tempos expressos em ciclos de máquina sem E/S (T) e com E/S (T*).

2 - As colunas referenciadas são colunas da Tabela A.2.

3 - Adicionar a todas as instruções a parcela de tempo referente às suas execuções na unidade aritmética ASTROM.

TABELA A.8

TEMPOS DE RESPOSTA DA UCP/ASTROP

EVENTO		TEMPO DE EXECUÇÃO		
INICIAL	FINAL	BÁSICO	ACESSO À PILHA	ACESSO AOS VETORES
Término do "Reset" da UCP/ASTROP	Início do "fetch" da próxima instrução	16T	-	2T*
"Trap" de violação dos limites da pilha, endereçamento ímpar de palavra, ASTROM, código de operação inválido.		12T	2T*	2T*
"Trap" de "timeout" no BASIS, de "breakpoint"		13T	2T*	2T*
"Trap" de erro de paridade no BASIS, atendimento de interrupção		14T	2T*	2T*

Observação: 1 - Tempos expressos em ciclos de máquina sem E/S (T) e com E/S (T*).

* = código de condição afetado pelo resultado da instrução.

• = código de condição afetado diretamente pela instrução.

TABELA B.1

INSTRUÇÕES DO COMPUTADOR ASTROP (VERSÕES 1.0 e 1.1 DO MICROPROGRAMA)

MNEMÔNICO	CÓDIGO DE OPERAÇÃO	OPERAÇÃO	CÓDIGOS DE CONDIÇÃO T1 T2 T3 C V Z S
		INSTRUÇÕES COM DOIS OPERANDOS	
ADD (B)	1/0 00100FFFFFDDDDD	(dst) ← (fnt) + (dst)	---****
DAC (B)	1/0 00101FFFFFDDDDD	(dst) ← (fnt) + (dst) + (c)	---****
SUB (B)	1/0 00110FFFFFDDDDD	(dst) ← (dst) - (fnt)	---****
DSB (B)	1/0 00111FFFFFDDDDD	(dst) ← (dst) - (fnt) - (c)	---****
CMP (B)	1/0 01000FFFFFDDDDD	(fnt) - (dst)	---****
AND (B)	1/0 01001FFFFFDDDDD	(dst) ← (fnt) [.] (dst)	----0**
OR (B)	1/0 01010FFFFFDDDDD	(dst) ← (fnt) [+] (dst)	----0**
(&)MSK (B)	1/0 01011FFFFFDDDDD	(dst) ← [-] (fnt) [.] (dst)	----0**
XOR (B)	1/0 01100FFFFFDDDDD	(dst) ← (fnt) [*] (dst)	----0**
(&)XNR (B)	1/0 01101FFFFFDDDDD	(dst) ← [-] [(fnt) [*] (dst)]	----0**
COC (B)	1/0 01110FFFFFDDDDD	[(fnt) [.] (dst)] [*] (fnt)	----0**
(&)CZC (B)	1/0 01111FFFFFDDDDD	[(fnt) [+] (dst)] [*] (fnt)	----0**
MOV (B)	1/0 10000FFFFFDDDDD	(dst) ← (fnt)	----0**
(&)EXC (B)	1/0 10001FFFFFDDDDD	(dst) ↔ (fnt)	----0**

(continua)

Tabela B.1 - Continuação

MNEMÔNICO	CÓDIGO DE OPERAÇÃO	OPERAÇÃO	CÓDIGOS DE CONDIÇÃO T1 T2 T3 C V Z S
DJZ	110010FFFFFDDDDD	(fnt) ← (fnt) - 1; (PC) ← (dst) se (fnt) resultar diferente de zero.	----***
JSB	110011FFFFFDDDDD	(tmp1) ← (fnt); (tmp2) ← (dst); (fnt) ← (PC); (SP) ← (SP) - 2; ((SP)) ← (tmp1); (PC) ← (tmp2).	-------
INSTRUÇÕES COM UM OPERANDO			
NEG (B)	1/0 0000000001DDDDD	(dst) ← - (dst)	---****
INC (B)	1/0 0000000010DDDDD	(dst) ← (dst) + 1	----***
DCR (B)	1/0 0000000011DDDDD	(dst) ← (dst) - 1	----***
(&)ICT (B)	1/0 0000000100DDDDD	(dst) ← (dst) + 2	----***
(&)DCT (B)	1/0 0000000101DDDDD	(dst) ← (dst) - 2	----***
CMT (B)	1/0 0000000110DDDDD	(dst) ← [-] (dst)	----0**
STZ (B)	1/0 0000000111DDDDD	(dst) ← 0	----010

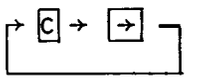
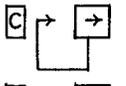
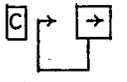
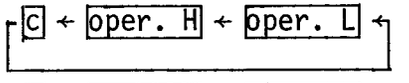
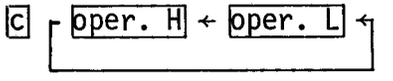
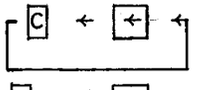
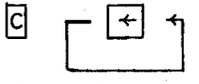
(continua)

Tabela B.1 - Continuação

MNEMÔNICO	CÓDIGO DE OPERAÇÃO	OPERAÇÃO	CÓDIGOS DE CONDIÇÃO T1 T2 T3 C V Z S
STO (B)	1/0 0000001000DDDDD	(dst) ← 1	----001
ABS (B)	1/0 0000001001DDDDD	(dst) ← - (dst), se (dst) < 0	---0***
ADC	1/0 0000001010DDDDD	(dst) ← (dst) + (c)	---****
SB (B)	1/0 0000001011DDDDD	(dst) ← (dst) - (c)	---****
CPZ (B)	1/0 0000001100DDDDD	(dst) - 0	---00**
(&) SXT (B)	1/0 0000001101DDDDD	(dst) ← 0, se código de cond. S é zero; (dst) ←-1, se código de cond. S é um.	----0*-
SLDL	10000010000DDDDD	[C] 0 → [oper. H] → [oper. L]	----0**
SDCL	10000010001DDDDD	0 → [C] → [oper. H] → [oper. L]	---00**
RDCL	10000010010DDDDD	→ [C] → [oper. H] → [oper. L] →	---*0**
SADL	10000010011DDDDD	[C] → [oper. H] → [oper. L]	----0**
RTDL	10000010100DDDDD	[C] → [oper. H] → [oper. L] →	----0**
SLD (B)	1/0 0000010101DDDDD	[C] 0 → [→]	----0**
SDC (B)	1/0 0000010110DDDDD	0 → [C] → [→]	---00**

(continua)

Tabela B.1 - Continuação

MNEMÔNICO	CÓDIGO DE OPERAÇÃO	OPERAÇÃO	CÓDIGOS DE CONDIÇÃO T1 T2 T3 C V Z S
RDC (B)	1/0 0000010111DDDDD		---*0**
SAD (B)	1/0 0000011000DDDDD		----0**
RTD (B)	1/0 0000011001DDDDD		----0**
SLEL	1000010000DDDDD	C oper. H ← oper. L ← Ø	----0**
SECL	1000010001DDDDD	C ← oper. H ← oper. L ← 0	---*0**
RECL	1000010010DDDDD		---*0**
RTEL	1000010011DDDDD		----0**
SLE (B)	1/0 0000100100DDDDD	C ← 0	----0**
SEC (B)	1/0 0000100101DDDDD	C ← ← 0	---*0**
REC (B)	1/0 0000100110DDDDD		---*0**
RTE (B)	1/0 0000100111DDDDD		----0**
SWB	1000010101 1DDDDD	"byte" H↔ "byte" L	----0**
RET	1000011000DDDDD	(PC) ← (dst); (dst)← ((SP)); (SP) ← (SP) + 2.	-------

(continua)

Tabela B.1 - Continuação

MNEMÔNICO	CÓDIGO DE OPERAÇÃO	OPERAÇÃO	CÓDIGOS DE CONDIÇÃO T1 T2 T3 C V Z S
JMP	10000110001DDDDD	(PC) ← (dst)	-------
SLP	10000110010DDDDD	(dst) ← (LP)	-------
CLP	10000110011DDDDD	(LP) ← (dst)	-------
SMK	10000110100DDDDD	(dst) ← (MK)	-------
CMK	10000110101DDDDD	(MK) ← (dst)	-------
DMI	10000110110DDDDD	(MK) ← (MK) [.] [-] (dst)	-------
MKI	10000110111DDDDD	(MK) ← (MK) [+] (dst), exceto bit 15	-------
CSPS	10000111000DDDDD	(SPS) ← (dst)	-------
SSPS	10000111001DDDDD	(dst) ← (SPS)	-------
SFGMB	00000111010DDDDD	(dst) ← códigos de condição do ASTROM	-------
INSTRUÇÕES COM ZERO OPERANDOS			
HALT	0000000000000000	"Halt" da UCP	-------
WAIT	0000000000000001	Espera interrupção	-------
NOP	0000000000000010	Nenhuma operação	-------
RST	0000000000000011	"Reset" da UCP	-------

(continua)

Tabela B.1 - Continuação

MNEMÔNICO	CÓDIGO DE OPERAÇÃO	OPERAÇÃO	CÓDIGOS DE CONDIÇÃO T1 T2 T3 C V Z S
EXS	1000000000000100	(temp1) ← ((SPS)); (SPS) ← (SPS) + 2; (temp2) ← ((SPS)); (SPS) ← (SPS) + 2; (SPS) ← (SPS) - 2; ((SPS)) ← (PC); (SPS) ← (SPS) - 2; ((SPS)) ← (PSW); (PSW) ← (temp1); (PC) ← (temp2).	.....
RTT	1000000000000101	(PSW) ← ((SPS)); (SPS) ← (SPS) + 2; (PC) ← ((SPS)); (SPS) ← (SPS) + 2.	.....
RTI	1000000000000101	(PSW) ← ((SPS)); (SPS) ← (SPS) + 2; (PC) ← ((SPS)); (SPS) ← (SPS) + 2.	.....

(continua)

Tabela B.1 - Continuação

MNEMÔNICO	CÓDIGO DE OPERAÇÃO	OPERAÇÃO	CÓDIGOS DE CONDIÇÃO T1 T2 T3 C V Z S
BTP	1000000000000111	(SPS) ← (SPS) - 2; ((SPS)) ← (PC); (SPS) ← (SPS) - 2; ((SPS)) ← (PSW); (PSW) ← ("FF98"H); (PC) ← ("FF9A"H).	.....
SRG	1000000000001000	(SP) ← (SP) - 2; ((SP)) ← (RF); (SP) ← (SP) - 2; ((SP)) ← (RE); (SP) ← (SP) - 2; ((SP)) ← (RD); (SP) ← (SP) - 2; ((SP)) ← (RC); (SP) ← (SP) - 2; ((SP)) ← (RB); (SP) ← (SP) - 2; ((SP)) ← (RA).	-------

(continua)

Tabela B.1 - Continuação

MNEMÔNICO	CÓDIGO DE OPERAÇÃO	OPERAÇÃO	CÓDIGOS DE CONDIÇÃO T1 T2 T3 C V Z S
CRG	1000000000001001	(RA) ← ((SP)); (SP) ← (SP) + 2; (RB) ← ((SP)); (SP) ← (SP) + 2; (RC) ← ((SP)); (SP) ← (SP) + 2; (RD) ← ((SP)); (SP) ← (SP) + 2; (RE) ← ((SP)); (SP) ← (SP) + 2; (RF) ← ((SP)); (SP) ← (SP) + 2.	-------
TRAPn	1000000000011nnn	(SPS) ← (SPS) - 2; ((SPS)) ← (PC); (SPS) ← (SPS) - 2; ((SPS)) ← (PSW); (PSW) ← ("FFAØ"H + 4 x nnn) (PC) ← ("FFA2"H + 4 x nnn)	.....

(continua)

Tabela B.1 - Continuação

MNEMÔNICO	CÓDIGO DE OPERAÇÃO	OPERAÇÃO	CÓDIGOS DE CONDIÇÃO T1 T2 T3 C V Z S
		INSTRUÇÕES COM PARÂMETROS	
DR	01110000PPPPPPPP	(PC)←(PC) + 2x"offset"	-------
DCO	01110001PPPPPPPP	(PC)←(PC) + 2x"offset" se C=1	-------
DCZ	01110010PPPPPPPP	(PC)←(PC) + 2x"offset" se C=0	-------
DVO	01110011PPPPPPPP	(PC)←(PC) + 2x"offset" se V=1	-------
DVZ	01110100PPPPPPPP	(PC)←(PC) + 2x"offset" se V=Ø	-------
DZO	01110101PPPPPPPP	(PC)←(PC) + 2x"offset" se Z=1	-------
DZZ	01110110PPPPPPPP	(PC)←(PC) + 2x"offset" se Z=0	-------
DSO	01110111PPPPPPPP	(PC)←(PC) + 2x"offset" se S=1	-------
DSZ	01111000PPPPPPPP	(PC)←(PC) + 2x"offset" se S=0	-------
DTAO	11111000PPPPPPPP	(PC)←(PC) + 2x"offset" se T1=1	-------
DTAZ	11111001PPPPPPPP	(PC)←(PC) + 2x"offset" se T1=0	-------
DTBO	11111010PPPPPPPP	(PC)←(PC) + 2x"offset" se T2=1	-------
DTBZ	11111011PPPPPPPP	(PC)←(PC) + 2x"offset" se T2=0	-------
DTCO	11111100PPPPPPPP	(PC)←(PC) + 2x"offset" se T3=1	-------
DTCZ	11111101PPPPPPPP	(PC)←(PC) + 2x"offset" se T3=0	-------
DGE	11110000PPPPPPPP	(PC)←(PC) + 2x"offset" se S[*] V=0 Desvio Relativo se ≧(número com sinal)	-------

(contin

Tabela B.1 - Continuação

MNEMÔNICO	CÓDIGO DE OPERAÇÃO	OPERAÇÃO	CÓDIGOS DE CONDIÇÃO T1 T2 T3 C V Z S
DLT	11110001PPPPPPPP	(PC) ← (PC) + 2x"offset" se S[*] V=1 Desvio Relativo se < (números com sinal)	-------
DGT	11110010PPPPPPPP	(PC) ← (PC) + 2x"offset" se Z[+] (S[*]V)=0 Desvio Relativo se > (números com sinal)	-------
DLE	11110011PPPPPPPP	(PC) ← (PC) + 2x"offset" se Z[+](S[*]V)=1 Desvio Relativo se ≦ (números com sinal)	-------
DHI	11110100PPPPPPPP	(PC) ← (PC) + 2x"offset" se C[+] Z=0 Desvio Relativo se > (números sem sinal)	-------
DLS	11110101PPPPPPPP	(PC) ← (PC) + 2x"offset" se C[+] Z=1 Desvio Relativo se ≦ (números sem sinal)	-------
DHS	01110010PPPPPPPP	(PC) ← (PC) + 2x"offset" se C=0 Desvio Relativo se ≧ (números sem sinal)	-------
DLO	01110001PPPPPPPP	(PC) ← (PC) + 2x"offset" se C=1 Desvio Relativo se < (números sem sinal)	-------
ERC	01111010XMMMMMMM	Entra Região condicional	.....
CFO	01111011XMMMMMMM	Faz códigos de condição iguais a 1	.....
CFZ	01111100XMMMMMMM	Faz códigos de condição iguais a 0	.....
CCC	01111101XMMMMMMM	Carrega códigos de condição	.....
CNM	01111110XXXXSSSS	Carrega nível mínimo de interrupção	.....

(continua)

Tabela B.1 - Continuação

MNEMÔNICO	CÓDIGO DE OPERAÇÃO	OPERAÇÃO	CÓDIGOS DE CONDIÇÃO T1 T2 T3 C V Z S
		INSTRUÇÕES PARA A UNIDADE ARITMÉTICA ASTROM	
SMIP	100010011AAFFFFF	(AC1)← (AC1) + (fnt) (inteiro)	-------
SMIM	000010011AAXXXBB	(AC1)← (AC1) + (AC2) (inteiro)	-------
SUIP	100010101AAFFFFF	(AC1)← (AC1) - (fnt) (inteiro)	-------
SUIM	000010101AAXXXBB	(AC1)← (AC1) - (AC2) (inteiro)	-------
MLIP	100010111AAFFFFF	(AC1)← (AC1) x (fnt) (inteiro)	-------
MLIM	000010111AAXXXBB	(AC1)← (AC1) x (AC2) (inteiro)	-------
DVIP	100011001AAFFFFF	(AC1)← (AC1) / (fnt) (inteiro)	-------
DVIM	000011001AAXXXBB	(AC1)← (AC1) / (AC2) (inteiro)	-------
SMPFP	100010010AAFFFFF	(AC1)← (AC1) + (fnt) (ponto flutuante)	-------
SUPFP	100010100AAFFFFF	(AC1)← (AC1) - (fnt) (ponto flutuante)	-------
SUPFM	000010100AAXXXBB	(AC1)← (AC1) - (AC2) (ponto flutuante)	-------
SMPFM	000010010AAXXXBB	(AC1)← (AC1) + (AC2) (ponto flutuante)	-------
MLPFP	100010110AAFFFFF	(AC1)← (AC1) x (fnt) (ponto flutuante)	-------
MLPFM	000010110AAXXXBB	(AC1)← (AC1) x (AC2) (ponto flutuante)	-------
DVPFP	100011000AAFFFFF	(AC1)← (AC1) / (fnt) (ponto flutuante)	-------
DVPFM	000011000AAXXXBB	(AC1)← (AC1) / (AC2) (ponto flutuante)	-------
NGIN	000011110AAXXXBB	(AC1)← - (AC2) (inteiro)	-------
NGPFP	100011010AAFFFFF	(AC1)← - (fnt) (ponto flutuante)	-------
NGPFM	000011010AAXXXBB	(AC1)← - (AC2) (ponto flutuante)	-------

(continua)

Tabela B.1 - Conclusão

MNEMÔNICO	CÓDIGO DE OPERAÇÃO	OPERAÇÃO	CÓDIGOS DE CONDIÇÃO T1 T2 T3 C V Z S
ABPF	000011100AAXXXBB	(AC1) ← módulo do (AC2) (ponto flutuante)	-------
ABSI	000011101AAXXXBB	(AC1) ← módulo do (AC2) (inteiro)	-------
CMPFP	100011011AAFFFFF	(AC1) - (fnt) (ponto flutuante)	-------
CMPFM	000011011AAXXXBB	(AC1) - (AC2) (ponto flutuante)	-------
IPFP	100010001AAFFFFF	(AC1) ponto flutuante ← (fnt) inteiro	-------
IPFM	000010001AAXXXBB	(AC1) ponto flutuante ← (AC2) inteiro	-------
PFIP	100010000AAFFFFF	(AC1) inteiro ← (fnt) ponto flutuante	-------
PFIM	000010000AAXXXBB	(AC1) ponto flutuante ← (AC2) inteiro	-------
CPRGI	000011111AAXXXBB	(AC1) ← (AC2) (inteiro)	-------
CRI	100011101AAFFFFF	(AC1) ← (fnt) (inteiro)	-------
CRPF	100011100AAFFFFF	(AC1) ← (fnt) (ponto flutuante)	-------
AZPF	100011110AADDDDD	(dst) ← (AC1) (ponto flutuante)	-------
AZI	100011111AADDDDD	(dst) ← (AC1) (inteiro)	-------

TABELA B.2

MODOS DE ENDEREÇAMENTO DO

COMPUTADOR ASTROP

REGISTRO ASSOCIADO	MODO	MODO DE ENDEREÇAMENTO
RA a RF	0	Registro direito
	1	Registro indireto
	2	Registro indexado
	3	Registro indexado indireto
SP	0	Ponteiro da pilha
	1	Ponteiro autodecrementado
	2	Ponteiro autoincrementado
	3	Topo da pilha
PC	0	Imediato
	1	Absoluto
	2	Relativo
	3	Relativo indireto