HIDRA – REFERÊNCIA DE USO

CONTEÚDO

MONTADOR

Formato de instruções e diretivas

Label: Mnemônico Argumentos ; Comentário

LABEL

Utilizada para nomear uma posição de memória, de forma que o programador possa referenciá-la como argumento de instruções sem saber o valor de seu endereço. Permite, por exemplo, a referência a variáveis e posições de código para desvio.
Não precisa estar na mesma linha que uma instrução ou diretiva. Não há distinção entre maiúsculas e minúsculas.

MNEMÔNICO

O mnemônico da instrução ou diretiva. Não há distinção entre maiúsculas e minúsculas.

ARGUMENTOS

Os argumentos da instrução ou diretiva, que podem ser valores decimais positivos ou negativos, valores hexadecimais (usando prefixo H) ou labels. Também podem ser o nome de um registrador, em certas máquinas e instruções.

COMENTÁRIO

Comentários de código são iniciados por ponto-e-vírgula, e seguem até o final da linha.

Diretivas do montador

ORG (Origin)

Sintaxe: **ORG** a
Posiciona a montagem das instruções subsequentes no endereço ‘a’ da memória.

DB (Define Byte)

Sintaxe: **DB** | **DB** a | **DB** ha | **DB** ‘a’
Reserva um byte na posição de montagem atual, opcionalmente inicializando-o com um valor ‘a’. Valores negativos são convertidos para complemento de dois. O prefixo ‘h’ indica valor hexadecimal. O valor pode também ser um caractere ASCII, cercando-o com aspas simples.
Exemplos: DB | DB 1 | DB -1 | DB h1A | DB ‘z’

DW (Define Word)

Sintaxe: **DW** | **DW** a | **DW** ha | **DW** ‘a’

Reserva uma palavra (16 bits, ou 2 bytes) na posição de montagem atual, com a mesma sintaxe de DB. Os 8 bits mais significativos são armazenados primeiro (big-endian).

DAB (Define Array of Bytes)

Sintaxe: **DAB** a | **DAB** a, a, a... | **DAB** a a a... | **DAB** ‘aaa’ | **DAB** [a]

Reserva uma sequência de um ou mais bytes. Os bytes podem ser valores declarados um a um, separados por vírgulas ou espaços, onde ‘a’ segue sintaxe de DB. Ainda, pode ser uma sequência de caracteres ASCII cercada por aspas simples.

Um número ‘a’ entre colchetes permite reservar ‘a’ valores sem inicializá-los.

Exemplos: DAB 1, -1, h1A, ‘z’ | DAB ‘abcde’ | DAB ‘’’ | DAB [20]

DAW (Define Array of Words)

Sintaxe: **DAW** a | **DAW** a, a, a... | **DAW** a a a... | **DAW** ‘aaa’ | **DAW** [a]

Reserva uma sequência de uma ou mais palavras (valores de 16 bits), com a mesma sintaxe de DAB. Os 8 bits mais significativos são armazenados primeiro (big-endian). Caracteres ASCII, portanto, são armazenados no segundo byte.

NEANDER

Modos de endereçamento

DIRETO

O valor do argumento ‘a’ em uma instrução representa o endereço onde o operando se encontra ou, em operações de jump, o endereço para qual o desvio será realizado.

Registradores

AC (Accumulator)

Registrador de 8 bits. Armazena o resultado de todas as operações lógicas e aritméticas. É sempre o primeiro operando de operações lógicas e aritméticas. Em operações de dois operandos, o segundo operando é retirado da memória.

PC (Program Counter)

Apontador de programa de 8 bits. Contém um endereço, que aponta para a instrução a ser executada no próximo passo da máquina.

Flags

N (Negative)

Interpreta o valor do acumulador como um valor em complemento de 2 (-128 a +127), sendo ativa quando o valor é negativo. Os valores de 255 a 128 representam a faixa de -1 a -128, sendo considerados negativos.

Z (Zero)

Se torna ativa quando o valor do acumulador é zero.

Instruções

NOP

Sintaxe: **NOP**
Nenhuma operação.

STA (Store accumulator)

Sintaxe: **STA** a
Armazena o valor do acumulador no endereço 'a'.

LDA (Load accumulador)

Sintaxe: **LDA** a
Carrega o valor no endereço 'a' para o acumulador.

ADD

Sintaxe: **ADD** a

Adiciona o valor no endereço 'a' ao acumulador. O bit de excesso (“vai-um”) é descartado. A operação funciona mesmo interpretando-se os valores como complemento de dois.

OR

Sintaxe: **OR** a

Realiza um 'ou' lógico entre cada bit de 'a' e o bit correspondente no acumulador.

AND

Sintaxe: **AND** a

Realiza um 'e' lógico entre cada bit de 'a' e o bit correspondente no acumulador.

NOT

Sintaxe: **NOT**

Inverte (complementa) o valor de cada um dos bits do acumulador.

JMP (Jump)

Sintaxe: **JMP** a

Desvia a execução para o endereço 'a' (desvio incondicional).

JN (Jump on Negative)

Sintaxe: **JN** a

Se a flag N estiver ativada (acumulador negativo), desvia a execução para o endereço 'a'.

JZ (Jump on Zero)

Sintaxe: **JZ** a

Se a flag Z estiver ativada (acumulador zerado), desvia a execução para o endereço 'a'.

HLT (Halt)

Sintaxe: **HLT**

Termina a execução.

AHMES

Ahmes é uma extensão da arquitetura Neander, com diversas flags e instruções adicionais.

Flags adicionais

V (Overflow)

Sinaliza a ocorrência estouro de representação (resultado fora da faixa de representação) na última operação de soma ou subtração realizada, interpretando os valores de operandos e resultado como complemento de dois (-128 a +127).

C (Carry)

Em operações de adição, sinaliza se houve carry (“vai-um”), interpretando os valores como inteiros positivos (0 a +255). Atua como bit extra em operações de rotação e shift.

B (Borrow)

Em operações de subtração, sinaliza se houve borrow (“empresta-um”), interpretando os valores como inteiros positivos (0 a +255).

Instruções adicionais

SUB (Subtract)

Sintaxe: **SUB** a
Subtrai o valor no endereço 'a' do acumulador.

JP (Jump on Positive)

Sintaxe: **JP** a
Se as flags N e Z estiverem desativadas (acumulador positivo), desvia a execução para o endereço 'a'.

JV (Jump on Overflow)

Sintaxe: **JV** a
Se a flag V estiver ativada (overflow), desvia a execução para o endereço 'a'.

JNV (Jump on Not Overflow)

Sintaxe: **JNV** a
Se a flag V estiver desativada (not overflow), desvia a execução para o endereço 'a'.

JNZ (Jump on Not Zero)

Sintaxe: **JNZ** a
Se a flag Z estiver desativada (acumulador diferente de zero), desvia a execução para o endereço 'a'.

JC (Jump on Carry)

Sintaxe: **JC** a

Se a flag C estiver ativada (carry), desvia a execução para o endereço 'a'.

JNC (Jump on Not Carry)

Sintaxe: **JNC** a

Se a flag C estiver desativada (not carry), desvia a execução para o endereço 'a'.

JB (Jump on Borrow)

Sintaxe: **JB** a

Se a flag B estiver ativada (borrow), desvia a execução para o endereço 'a'.

JNB (Jump on Not Borrow)

Sintaxe: **JNB** a

Se a flag B estiver desativada (not borrow), desvia a execução para o endereço 'a'.

SHR (Shift Right)

Sintaxe: **SHR**

Reliza shift lógico dos bits do acumulador para a direita, passando o estado do bit menos significativo para a flag C (carry) e preenchendo o bit mais significativo com 0.

SHL (Shift Left)

Sintaxe: **SHL**

Reliza shift lógico dos bits do acumulador para a esquerda, passando o estado do bit mais significativo para a flag C (carry) e preenchendo o bit menos significativo com 0.

ROR (Rotate Right)

Sintaxe: **ROR**

Realiza rotação para a esquerda dos bits do acumulador, incluindo a flag C (carry) como um bit.

ROL (Rotate Left)

Sintaxe: **ROL**

Realiza rotação para a direita dos bits do acumulador, incluindo a flag C (carry) como um bit.

RAMSES

Ramses também é uma extensão da arquitetura Neander, porém, com instruções foram adaptadas para suportar múltiplos registradores.

Modos de endereçamento

A presença de prefixos ou sufixos para o argumento ‘a’ é capaz de alterar como uma instrução interpreta e busca seu operando em memória.

DIRETO

Sintaxe: a (somente valor/label)
O valor do argumento ‘a’ representa o endereço onde o operando se encontra ou, em operações de jump, o endereço para qual o desvio será realizado.

INDIRETO

Sintaxe: a**,I** (valor/label com sufixo ,I)
O valor do argumento ‘a’ representa um endereço que por sua vez contém o endereço direto.

IMEDIATO

Sintaxe: **#**a (valor/label com prefixo #)
O valor do argumento ‘a’ não representa um endereço, e sim o valor imediato que deve ser carregado no registrador (instrução LDR) ou utilizado em operações aritméticas.

INDEXADO

Sintaxe: a**,X** (valor/label com sufixo ,x)
Endereçamento direto com deslocamento. A soma dos valores de ‘a’ e do registrador X representam o endereço direto.

Registradores

A, B

Registradores de uso geral de 8 bits.

X

Registrador de uso geral de 8 bits, utilizado também pelo modo de endereçamento indexado para determinar o deslocamento.

PC (Program Counter)

Apontador de programa de 8 bits. Contém um endereço, que aponta para a instrução a ser executada no próximo passo da máquina.

Flags

N (Negative)
Indica se o valor no último registrador utilizado por uma operação lógica ou aritmética é negativo (em complemento de dois).

Z (Zero)
Indica se o valor no último registrador utilizado por uma operação lógica ou aritmética é zero.

C (Carry)
Além de indicar carry (“vai-um”) em operações de adição, a flag indica a não-ocorrência de borrow (“empresta-um”) em operações de subtração. Atua como bit extra em operações de shift.

Instruções

NOP
Sintaxe: **NOP**
Nenhuma operação.

STR (Store Register)
Sintaxe: **STR** r a
Armazena o valor do registrador 'r' no endereço 'a'.

LDR (Load Register)
Sintaxe: **LDR** r a
Carrega o valor no endereço 'a' para o registrador 'r'.

ADD
Sintaxe: **ADD** r a
Adiciona o valor no endereço 'a' ao registrador 'r'.

OR
Sintaxe: **OR** r a
Realiza um 'ou' lógico entre cada bit de 'a' e o bit correspondente no registrador 'r'.

AND
Sintaxe: **AND** r a
Realiza um 'e' lógico entre cada bit de 'a' e o bit correspondente no registrador 'r'.

NOT
Sintaxe: **NOT** r
Inverte (complementa) o valor dos bits do registrador 'r'.

SUB (Subtract)

Sintaxe: **SUB** r a

Subtrai o valor no endereço 'a' do registrador 'r'.

JMP (Jump)

Sintaxe: **JMP** a

Desvia a execução para o endereço 'a' (desvio incondicional).

JN (Jump on Negative)

Sintaxe: **JN** a

Se a flag N estiver ativada (acumulador negativo), desvia a execução para o endereço 'a'.

JZ (Jump on Zero)

Sintaxe: **JZ** a

Se a flag Z estiver ativada (acumulador zerado), desvia a execução para o endereço 'a'.

JC (Jump on Carry)

Sintaxe: **JC** a

Se a flag C estiver ativada (carry), desvia a execução para o endereço 'a'.

JSR (Jump to Sub-Routine)

Sintaxe: **JSR** a

Desvia para sub-rotina, armazenando o valor atual de PC em 'a' e desviando a execução para o endereço 'a' + 1.

NEG (Negate)

Sintaxe: **NEG** r

Troca o sinal do valor em complemento de 2 do registrador 'r' de positivo para negativo e vice-versa.

SHR (Shift Right)

Sintaxe: **SHR** r

Realiza shift lógico dos bits do registrador 'r' para a direita, passando o estado do bit menos significativo para a flag C (carry) e preenchendo o bit mais significativo com 0.