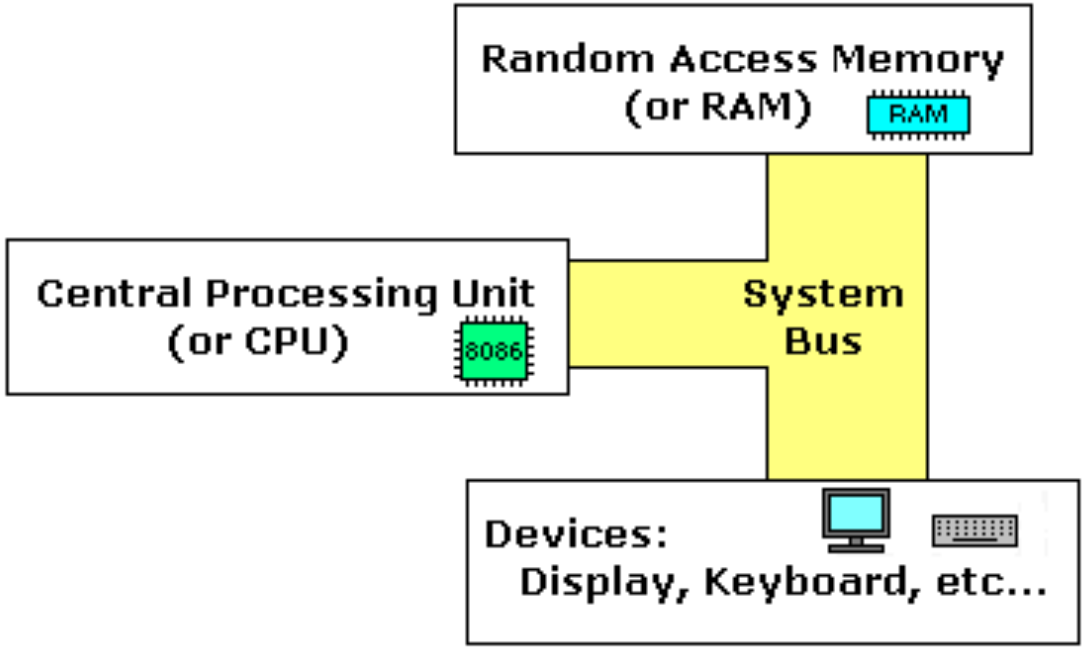
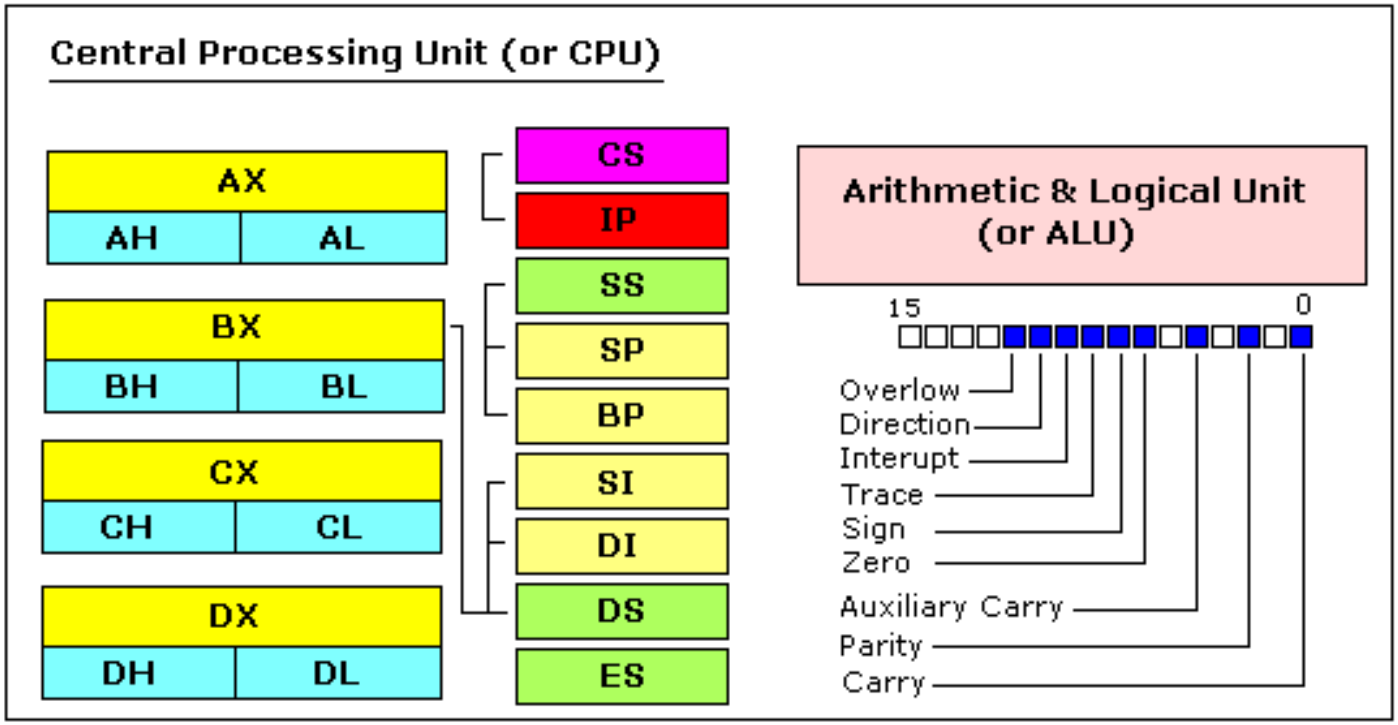
Linguagem de Montagem Assembly 8086

Prof. Francisco Isidro

Assembly – Modelo de Computador



Por dentro da CPU



Registradores

- 8086 possui 8 registradores de propósito geral
  - AX – acumulador (dividido em AH/AL)
  - BX – endereço base (dividido em BH/BL)
  - CX – contador (dividido em CH/CL)
  - DX – registrador de dados (dividido em DH/DL)
  - SI – “Source Index” – Índice de origem
  - DI – “Destination Index” – Índice de Destino
  - BP – Ponteiro Base
  - SP – Ponteiro de Pilha

Registradores

- Localizados dentro da CPU
  - Muito mais rápido que a memória
- Apesar dos nomes, quem define sua real função é o programador
- Tamanho do registrador
  - 16 bits
  - 2 partes de 8 bits cada
  - Exemplo
    - AX= 0011000000111001b
      - AH=00110000b and AL=00111001b

Outros Registradores

- Registradores de acesso a segmentos de memória
  - CS – Code Segment
    - Indica onde o código do programa está localizado (primeira posição)
  - DS – Data Segment
    - Local onde as variáveis estão definidas
  - ES – Extra Segment
    - Segmento extra para uso geral
  - SS – Stack Segment
    - Segmento para armazenar a pilha (geralmente usado em chamadas de funções)
- Registradores de uso especial
  - IP – Instruction Pointer
  - Flags
    - indicam o estado do microprocessador

Acesso à Memória

- Pode-se utilizar os seguintes registradores
  - BX
  - SI
  - DI
  - BP
- Cada endereço de memória contém 1 palavra
  - 2 bytes
  - Idéia de caber nos registradores

Acesso à Memória

- Operador [ ]
  - Indica conteúdo de uma determinada posição
- Calculando um endereço real
  - Suponha
    - DS = 100
    - BX = 30
    - SI = 70
  - Calcule [BX + SI] + 25
    - Endereço = 100 * 16 + 30 + 70 + 25 = 1725

Acesso à Memória

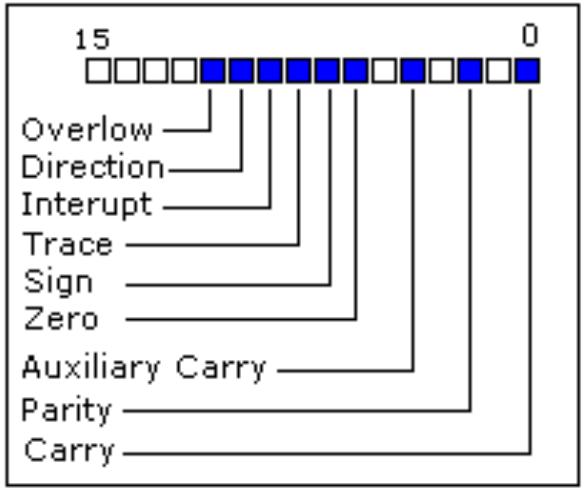
- Valores nos registradores DS,CS,ES,SS
  - Chamados de Segmentos
- Valores nos registradores BX,SI,DI,BP
  - Chamados de Offset – deslocamento
- Exemplo
  - DS = 1234h
  - SI = 7890h
- Gravado como 1234:7890
- Endereço Físico
  - 1234*10h + 7890h = 19BD0h

Notação!!

- h indica *hexadecimal*
- Cuidado nos cálculos
  - 7h = 7 (decimal)
  - 70h = 112 (decimal)

Flags de Registradores

- Indicam estado e podem alterar o funcionamento da máquina
  - OF – Overflow
  - DF – Direction
  - SF – Sign (sinal)
  - PF – Paridade
  - IF – Interrupção
  - CF – Carry
  - ZF – Zero
  - AF – Carry Auxiliar



Instruções de Movimentação

- MOV
  - Copia o segundo operando (origem) ao primeiro (destino)
- Sintaxe
  - MOV X,Y
    - Y origem
    - X destino
  - Origem pode ser: valor imediato, registrador ou posição de memória
  - Destino pode ser: registrador ou posição de memória

Instruções de Movimentação

- Tipos de operandos MOV
  - MOV REG, memory
  - MOV memory, REG
  - MOV REG, REG
  - MOV memory, immediate
  - MOV REG, immediate
- REG
  - AX, BX, CX, DX, AH, AL, BL, BH, CH, CL, DH, DL, DI, SI, BP, SP.
- Memory
  - [BX], [BX+SI+7], variable, etc...
- Immediate
  - 5, -24, 3Fh, 10001101b, etc...

Exemplo

```
MOV AX, B800h       ; atribui a AX o valor hexa B800h
MOV DS, AX          ; copia o valor de AX para DS
MOV CL, 'A'         ; atribui a CL o valor ASCII do
                      caractere 'A', que é 41h
MOV CH, 1101_1111b ; atribui a CH um valor binário.
MOV BX, 15Eh        ; atribui a BX o valor hexa 15E
MOV [BX], CX        ; copia o conteúdo de CX para o
                      endereço de memória B800:015E
RET                 ; termina o programa e retorna ao SO
```

- O que faz esse código?
  - Copia o valor ‘A’ para um endereço de memória que corresponde à memória de vídeo
  - Desse modo, imprime no monitor o caractere.

Tente fazer...

- Mude o código do exemplo anterior para imprimir a mensagem “Hello”
- Mude o código que você construiu para imprimir a seguinte mensgem:

H
e
l
l
o

Operações Aritméticas

- ADD X, Y
  - X = X+Y
    - REG, memory
    - memory, REG
    - REG, REG
    - memory, immediate
    - REG, immediate
  - Exemplo

```
MOV AL, 5   ; AL = 5
ADD AL, -3  ; AL = 2
RET
```

Operações Aritméticas

- SUB X, Y
  - X = X – Y
    - REG, memory
    - memory, REG
    - REG, REG
    - memory, immediate
    - REG, immediate
  - Exemplo

```
MOV AL, 5
SUB AL, 1        ; AL = 4
RET
```

Operações Aritméticas

- MUL X
  - Quando o operando é um Byte (8 bits)
    - AX = AL*operando
  - Quando o operando é uma palavra (16 bits)
    - (DX AX) = AX*operando
  - Exemplos

```
MOV AL, 200   ; AL = 0C8h
MOV BL, 4
MUL BL        ; AX = 0320h (800)
RET
```

Operações Aritméticas

- DIV X
  - Quando o operando é um Byte (8 bits)
    - AL = AX / operando
    - AH = Módulo (remainder)
  - Quando o operando é uma Palavra (16 bits)
    - AX = (DX AX) / operando
    - DX = Módulo (remainder)
  - Exemplo

```
MOV AX, 203   ; AX = 00CBh
MOV BL, 4
DIV BL        ; AL = 50 (32h), AH = 3
RET
```

Operações Aritméticas

- CMP X, Y
  - X – Y
    - REG, memory
    - memory, REG
    - REG, REG
    - memory, immediate
    - REG, immediate
  - Altera um flag de registrador (ZF) para 1 (ZF = 1), se forem iguais
- Exemplo

```
MOV AL, 5
MOV BL, 5
CMP AL, BL  ; AL = 5, ZF = 1 (so equal!)
RET
```

Operações Aritméticas

- INC X
  - X = X+1
    - REG
    - Memory
  - Exemplo

```
MOV AL, 4
INC AL          ; AL = 5
RET
```

Operações Aritméticas

- DEC X
  - X = X – 1
    - REG
    - Memory
  - Exemplo

```
MOV AL, 255    ; AL = 0FFh (255 or -1)
DEC AL         ; AL = 0FEh (254 or -2)
RET
```

Operações Aritméticas

- NEG
  - Realiza o complemento de 2 a um operando (inverte os bits e soma 1)
  - NEG X
    - REG
    - Memory
  - Exemplo

```
MOV AL, 5      ; AL = 05h
NEG AL         ; AL = 0FBh (-5)
NEG AL         ; AL = 05h (5)
RET
```

Operações Lógicas

- AND X, Y
  - X = X AND Y (bit a bit)
    - REG, memory
    - memory, REG
    - REG, REG
    - memory, immediate
    - REG, immediate
  - Exemplo

```
MOV AL, 'a'              ; AL = 01100001b
AND AL, 11011111b  ; AL = 01000001b  ('A')
RET
```

Operações Lógicas

- OR X, Y
  - X = X OR Y (bit a bit)
    - REG, memory
    - memory, REG
    - REG, REG
    - memory, immediate
    - REG, immediate
  - Exemplo

```
MOV AL, 'A'             ; AL = 01000001b
OR AL, 00100000b  ; AL = 01100001b  ('a')
RET
```

Operações Lógicas

- NOT
  - Inverte os bits de um operando
  - Se for 0, torna-o 1
  - Se for 1, torna-o 0
    - REG
    - Memory
- Exemplo

```
MOV AL, 00011011b
NOT AL   ; AL = 11100100b
RET
```

Operações Lógicas

- SHL X, Y
  - Deslocamento à esquerda
  - Insere zeros à direita
    - memory, immediate
    - REG, immediate
    - memory, CL
    - REG, CL
  - Se houver Carry, seta o flag CF para 1
  - Exemplo
    - MOV AL, 11100000b
    - SHL AL, 1 ; AL = 11000000b, CF=1.
    - RET

Operações Lógicas

- SHR X, Y
  - Desloca X à direita
  - Insere zeros à esquerda
    - memory, immediate
    - REG, immediate
    - memory, CL
    - REG, CL
  - Se houver Carry, seta o flag CF para 1
  - Exemplo
    - MOV AL, 00000111b
    - SHR AL, 1 ; AL = 00000011b, CF=1.
    - RET

Operações Lógicas

- ROL
  - Rotaciona todos os bits à esquerda
  - O bit de CF é colocado à direita
    - memory, immediate
    - REG, immediate
    - memory, CL
    - REG, CL
  - Exemplo
    - MOV AL, 1Ch ; AL = 00011100b
    - ROL AL, 1 ; AL = 00111000b, CF=0.
    - RET

Operações Lógicas

- ROR
  - Rotaciona todos os bits à direita
  - O bit de CF é colocado à esquerda
    - memory, immediate
    - REG, immediate
    - memory, CL
    - REG, CL
  - Exemplo
    - MOV AL, 1Ch ; AL = 00011100b
    - ROR AL, 1 ; AL = 00001110b, CF=0.
    - RET

XOR

- XOR X, Y
  - X = X XOR Y
    - REG, memory
    - memory, REG
    - REG, REG
    - memory, immediate
    - REG, immediate
  - Exemplo

```
MOV AL, 00000111b
XOR AL, 00000010b      ; AL = 00000101b
RET
```

Operações de Desvio

- JMP “label”
  - Desvio incondicional
  - Sempre desvia o processamento para um determinado “label”
- Exemplo

```
    MOV AL, 5
    JMP label1     ; jump over 2 lines!
    PRINT 'Not Jumped!'
    MOV AL, 0
 label1:
    PRINT 'Got Here!'
    RET
```

Operações de Desvio

- JZ “label”
  - Verifica se o flag ZF é 1. Se for verdadeiro, desvia
  - Usado sempre em combinação com o operador de comparação
- Exemplo

```
    MOV AL, 5
    CMP AL, 5
    JZ  label1
    PRINT 'AL is not equal to 5.'
    JMP exit
label1:
    PRINT 'AL is equal to 5.'
exit:
    RET
```

Operações de Desvio

- JNZ ‘Label’
  - Desvia se o flag ZF não for 1, ou seja, se a comparação entre 2 números for diferente
  - Exemplo

```
    MOV AL, 00000111b   ; AL = 7
    CMP  AL, 0          ; just set flags.
    JNZ label1
    PRINT 'zero.'
    JMP exit
label1:
    PRINT 'not zero.'
exit:
    RET
```

Operações de Desvio

- RET
  - Retorna de uma função chamada
  - Se for utilizada na função principal, retorna ao Sistema Operacional
  - Senão, retorna para a próxima instrução após a chamada da função (ver mais adiante)