

Linguagens de Montagem

DEMAC – Departamento de Estatística
Matemática Aplicada e Computação
UNESP – Rio Claro

**Prof. Daniel Carlos Guimarães
Pedronette**

Aula 2.

Introdução a Linguagem Assembly

Linguagem Assembly

- Programas em Assembly são criados utilizando três diferentes tipos de *statements*:

Instruções:

Dizem ao processador o que fazer
OpCode e operandos
Geram linguagem de máquina

Diretivas

Pseudo ops
Não geram linguagem de máquina

Macros

Conjunto de statements

Linguagem Assembly

- Formato dos statements:

```
[label]      mnemonic      [operands]      [;comment]
```

Exemplo:

Instrução:

```
repeat:      inc          result      ;increment result by 1
```

(result: endereço de memória)

Linguagem Assembly

- Formato dos statements:

```
[label]      mnemonic      [operands]      [;comment]
```

Exemplo:

Diretiva:

```
CR      EQU      0DH      ;carriage-return character
```

(Diretiva EQU atribui um valor – ASCII carriage return – para uma constante CR)

Linguagem Assembly

- Alocação de Memória
 - Em linguagens de alto nível (C por exemplo) a alocação de memória é feita de acordo com os tipos de variáveis utilizadas no programa:
 - Essas declarações não só reservam a quantidade necessária de espaço como identificam como o padrão de bits deve ser interpretado

```
char        response;    /* allocates 1 byte */
int         value;       /* allocates 4 bytes */
float       total;       /* allocates 4 bytes */
double      temp;        /* allocates 8 bytes */
```

Linguagem Assembly

- Alocação de Memória
Através da diretiva “define”:

```
[variable-name] define-directive initial-value [,initial-value],...
```

DB	Define Byte	; allocates 1 byte
DW	Define Word	; allocates 2 bytes
DD	Define Doubleword	; allocates 4 bytes
DQ	Define Quadword	; allocates 8 bytes
DT	Define Ten Bytes	; allocates 10 bytes

Linguagem Assembly

- Alocação de Memória

Exemplos de alocação de byte (DB):

```
sorted    DB    'y'
```

Equivalente a:

```
sorted    DB    79H
```

```
sorted    DB    1111001B
```


Linguagem Assembly

- Alocação de Memória

Exemplos de alocação de word (DW):
Inicializado com valor 25159

16-bit binary equivalent (6247H)

Little Endian:

address:	x	x+1
contents:	47	62

Linguagem Assembly

- Alocação de Memória

Exemplos de alocação de double (DD):
Inicializado com valor :

542803535 (205A864FH)

Little Endian:

address:	x	x+1	x+2	x+3
contents:	4F	86	5A	20

Linguagem Assembly

- Range de Alocação das diretivas:

Directive	Valid range
DB	-128 to 255 (i.e., -2^7 to $2^8 - 1$)
DW	$-32,768$ to $65,535$ (i.e., -2^{15} to $2^{16} - 1$)
DD	$-2,147,483,648$ to $4,294,967,295$ (i.e., -2^{31} to $2^{32} - 1$) or a short floating-point number (32 bits)
DQ	-2^{63} to $2^{64} - 1$ or a long floating-point number (64 bits)

Linguagem Assembly

- Alocação de Memória:
Sem inicialização de dados:
Diretiva “reserve”

RESB	Reserve a Byte
RESW	Reserve a Word
RESQ	Reserve a Doubleword
RESQ	Reserve a Quadword
REST	Reserve Ten Bytes

Linguagem Assembly

- Alocação de Memória:
Exemplos de diretiva reserva:
Espaço de 1 byte
Espaço para array com 100 words
Espaço para 1 double com 1 double

```
response    RESB    1
buffer      RESW   100
total       RESD    1
```

Estrutura dos programas Assembly

- Seções:
 - .data
 - .bss
 - .text

Estrutura dos programas Assembly

- A seção “.data”
 - Porção do código para declarar dados inicializados
 - Em outras palavras, definição de constantes
 - Aqui, podemos usar a diretiva “define”

```
section .data
    message:    db 'Hello world!'      ; Declare message to contain the bytes 'Hello world!' (without quotes)
    msglength:  equ 12                 ; Declare msglength to have the constant value 12
    buffersize: dw 1024                ; Declare buffersize to be a word containing 1024
```

Estrutura dos programas Assembly

- A seção “.bss”
 - Seção para declarar variáveis
 - Espaço de memória não inicializado
 - Aqui podemos usar a diretiva “reserve”

```
section .bss
    filename:    resb    255        ; Reserve 255 bytes
    number:      resb     1        ; Reserve 1 byte
    bignum:      resw     1        ; Reserve 1 word (1 word = 2 bytes)
    realarray:   resq    10        ; Reserve an array of 10 reals
```


Estrutura dos programas Assembly

- A seção “.text”
 - Seção onde o código é efetivamente escrito
 - Deve iniciar com “global _start” que indica onde o programa inicia:

```
section .text
    global _start

_start:
    pop    ebx        ; Here is the where the program actually begins
    .
    .
    .
```

System Calls

- Tipos de chamada:
 - Para encerrar o programa
 - Para exibir algo no vídeo
 - (...)
- Exemplo:

```
mov    eax,1      ; The exit syscall number
mov    ebx,0      ; Have an exit code of 0
int    80h        ; Interrupt 80h, the thing that pokes the kernel and says, "Yo, do this"
```

Montadores

- NASM
Netwide Assembly
Portabilidade
Intel sintaxe
- GAS
GNU Assembly
AT&T sintaxe
- Sintaxes diferentes
Mesma arquitetura, mesma semântica!

Primeiros Programas

- Program.asm (NASM)
- Program.as (GAS)
 - Montar
 - Linkar
 - Executar!
 - Funcionou?

Primeiros Programas

- Hello.asm (NASM)
- Hello.as (GAS)
 - Montar
 - Linkar
 - Executar!
 - Funcionou?

Primeiros Programas

- Arquivos .lst
- Visualizando o resultado
 - Assembly
 - Código de Máquina