Linguagens de Montagem

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

Prof. Daniel Carlos Guimarães Pedronette

Aula1. Introdução

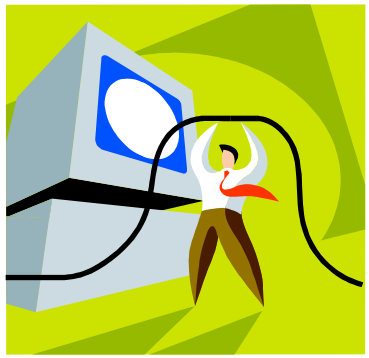
Linguagens de Montagem: Introdução, Motivações

Agenda

- Visões de um Sistema de Computação
- Diferentes Níveis de Abstração
  - Linguagens de Alto Nível
  - Assembly
  - Linguagem de Máquina
- Motivação: Por que Assembly?
- Revisão da Aula
- Exercícios

Visões de um Sistema de Computação

Planilhas, Editores de Texto, Navegadores

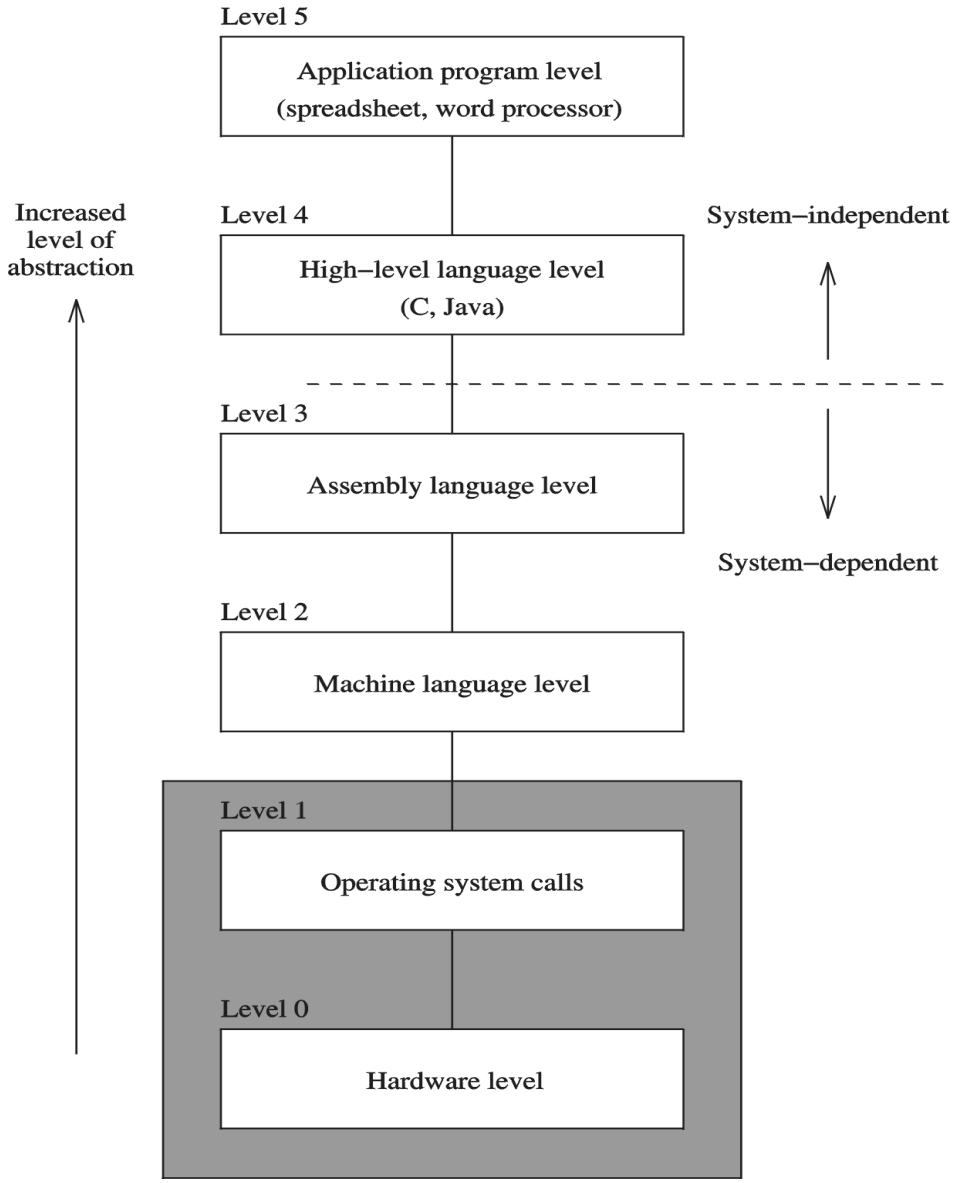


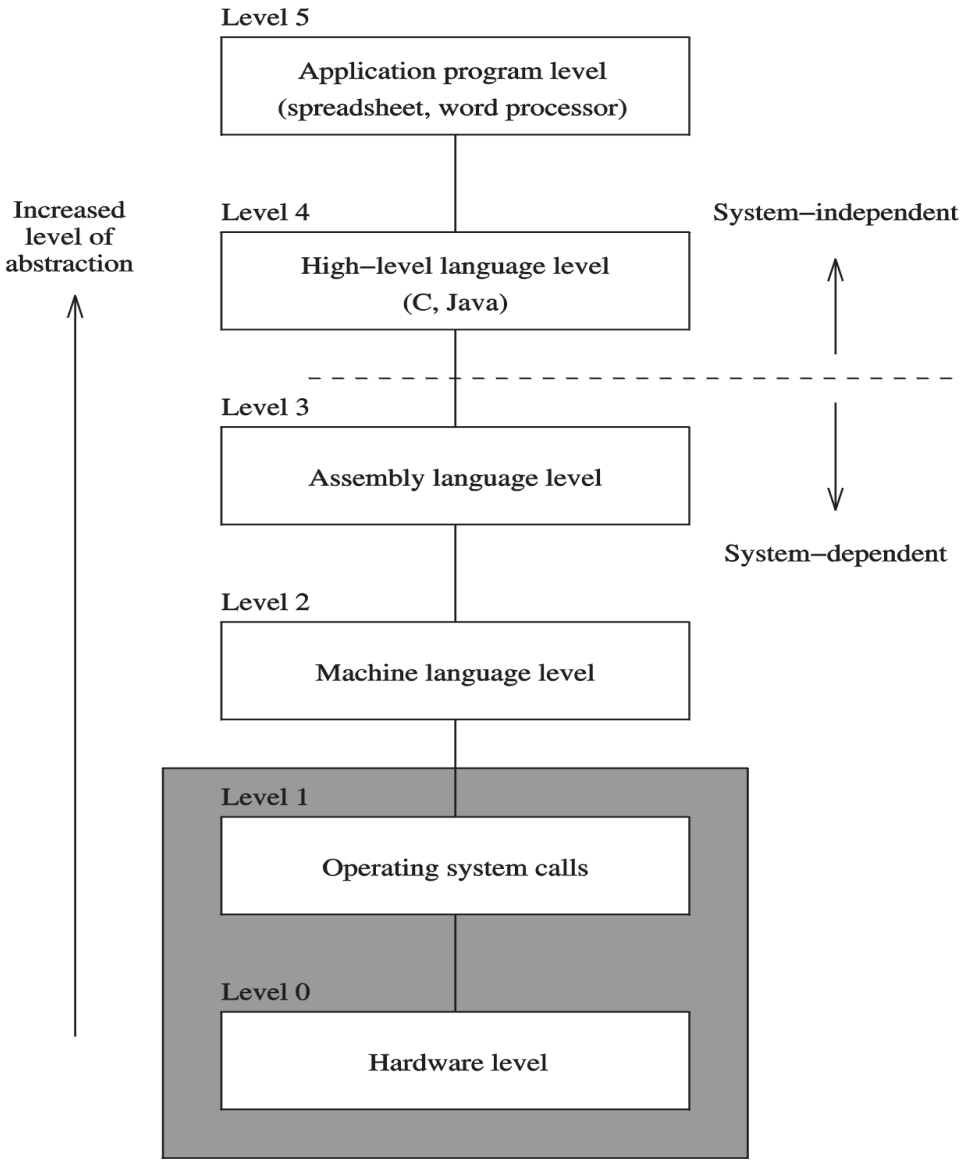
Desenvolvimento em Linguagens de Alto Nível, C/C++, Java

Linguagem de Montagem Assembly

Diferentes visões de usuário

- Visão do usuário:
  - Depende diretamente do grau de abstração provido pelo sofware de apoio.
- Considerando 5 níveis na hierarquia:
  - Nível 5: Aplicações
  - Nível 4: Linguagens de Alto Nível
  - Nível 3: Linguagem de Montagem
  - Nível 2: Linguagem de Máquina
  - Nível 1: Hardware e S.O.


Level 5
Application program level
(spreadsheet, word processor)
Increased level of abstraction
Level 4
High–level language level
(C, Java)
System–independent
Level 3
Assembly language level
System–dependent
Level 2
Machine language level
Level 1
Operating system calls
Level 0
Hardware level



- Movendo-se para alto da hierarquia, o usuário é poupado dos detalhes de baixo nível;
- Níveis superiores (top 2) são independentes do hardware;
- Níveis inferiores são dependentes do hardware;
- Assembly e linguagem de máquina são específicas para um processador em particular
- Correspondência um-para-um entre linguagem Assembly e linguagem de máquina

Linguagem Assembly

Linguagem Assembly é definida de acordo (ou diretamente influenciada) pelo conjunto de instruções e arquitetura do processador.

Interface disponível ao programador

"Low-level language"

Cada instrução executa tarefas próximas da máquina, principalmente se comparada com linguagens de alto nível.

Linguagem Assembly

- Correspondência entre Assembly e linguagem de máquina
  - Para a maior parte das instruções Assembly há uma equivalente em linguagem de máquina
  - *Assembler* traduz instruções em linguagem Assembly para linguagem de máquina

Taxonomia:

- *Assembly*
  - Linguagem de mnemônicos *próxima* da linguagem de máquina.
- *Assembler*
  - Programa/Aplicativo que traduz um código em linguagem Assembly para linguagem de máquina.

Assemblers

- Exemplos de Assemblers:

- **NASM (Netwide Assembler)**
- **GAS (GNU Assembler)**
- MASM (Microsoft Assembler)
- TASM (Borland Turbo Assembler)

Assemblers

- Diferença entre Assemblers:
  - Em geral, diferença está limitada à Sintaxe
  - Semântica é a mesma:
    - Significado do programa e instruções utilizadas estão relacionados à arquitetura e não ao Assembler

Linguagem Assembly

- Some example assembly language instructions:

```
inc    result
mov    class_size,45
and    mask1,128
add    marks,10
```

- Note que:
  - Linguagem Assembly utiliza mnemônicos para operações
  - inc for increment
  - mov for move (i.e., copy)
- Assembly utiliza instruções de baixo nível
  - Não é possível utilizar instruções como:

```
mov    marks, value
```

Linguagem Assembly

- Correspondência de instruções:
  - Algumas instruções de alto nível simples podem ser expressadas por uma correspondente Assembly

Assembly Language		C
inc	result	result++;
mov	class_size,45	class_size = 45;
and	mask1,128	mask1 &= 128;
add	marks,10	marks += 10;

Linguagem Assembly

- Correspondência de instruções:
  - Maioria das instruções de alto nível requerem várias instruções Assembly

C	Assembly Language
`size = value;`	`mov AX,value` `mov size,AX`

`sum += x + y + z;`	`mov AX,sum`
	`add AX,x`
	`add AX,y`
	`add AX,z`
	`mov sum,AX`

Linguagem Assembly

- Legibilidade do código Assembly muito superior à linguagem de máquina.

Assembly Language		Machine Language (in Hex)
inc	result	FF060A00
mov	class_size,45	C7060C002D00
and	mask,128	80260E0080
add	marks,10	83060F000A

Linguagem Assembly

- Linguagem de máquina:
  - Sequências de 0 e 1 !
  - Qual o significado de cada 0 e 1?
  - Nós poderíamos programar linguagem de máquina.
  - Ou seja, poderíamos “programar em 0 e 1”
  - Em geral, linguagem de máquina visualizadas utilizando Hexa.
  - Por que?

Por que Assembly?

- Por que utilizar Linguagem Assembly e não diretamente linguagem de máquina?
- Há alguma perda em utilizar Assembly e não diretamente linguagem de máquina?

Linguagens de Alto Nível

- Desenvolvimento é mais rápido:
  - *"High-level instructions"*
  - Menos instruções requeridas para codificação:
- Manutenção é mais simples
- Portabilidade dos programas
  - Contém poucos detalhes dependentes de máquina
  - Podem ser usados com poucas (ou nenhuma) modificações em diferentes tipos de máquina
  - Programas em Assembly não são portáveis

Por que Assembly?

- Por que estudar Assembly?
  - Alguém precisa traduzir o código de alto nível em linguagem de máquina...
  - Compilador! *Quem cria o compilador?*
- Para construir um compilador, é necessário, além de verificar e analisar sua estrutura, produzir código Assembly correspondente.

Por que Assembly?

- Qual o papel do Assembly no processo de compilação de linguagens de alto nível?
  - Em geral:
  - Escrevemos um programa em linguagem de alto nível (Linguagem C, por exemplo)
  - Utilizamos um “compilador”
  - Obtemos um arquivo executável!
  - Mas o que realmente acontece nesse processo?

Por que Assembly?

- Compilador:
- Entrada: programa em linguagem de alto nível
- Análise léxica
- Análise sintática
- Análise semântica
- (...)
- Saída: programa em Assembly!

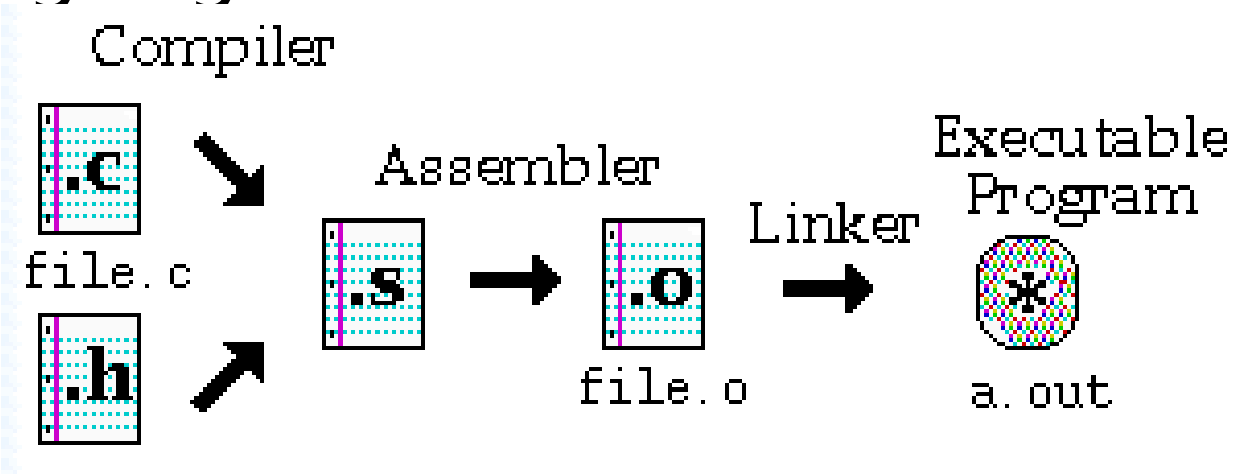
Por que Assembly?

- Assembler:
- Entrada: programa em Assembly
  - Utilização de mnemônicos
- Saída: linguagem de máquina (programa objeto)

- Linker
- Entrada: programa objeto
  - Ligação de bibliotecas, programas objetos
- Saída: executável

Por que Assembly?

- Qual o papel do Assembly no processo de compilação de linguagens de alto nível.



Por que Assembly?

- Otimização:
  - Por trabalhar diretamente com instruções de baixo nível, é possível produzir código mais eficiente e menores em diversas situações.
  - Compiladores são otimizados:
    - Otimização consiste em escrever código em Assembly, não compilar manualmente

Por que Assembly?

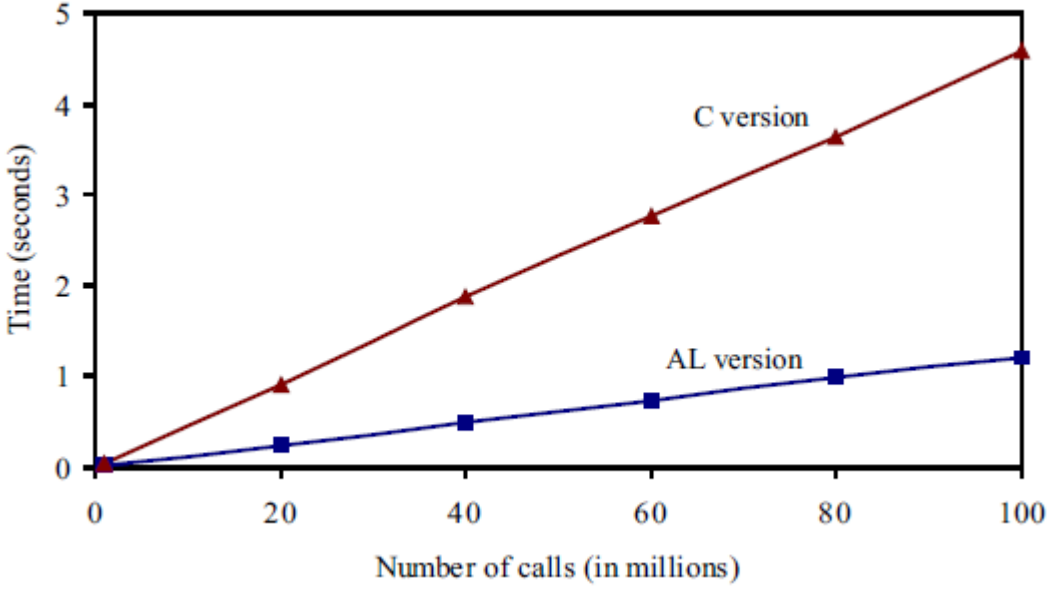
- Aplicações que necessitam de algumas vantagens da linguagem Assembly são indicadas a utilizá-las:
  - Eficiência
    - Programas escritos em Assembly podem ser mais rápidos em em linguagens de alto nível
  - Acesso direto ao hardware
  - Espaço

Por que Assembly?

- Eficiência
  - *Time-convenience* (desejável)
    - Aplicações Gráficas
  - *Time-critical* (imprescindível)
    - Aviação
    - Processos de controle industrial
    - Robótica

Por que Assembly?

- Comparação de um procedimento de multiplicação em C em Assembly:



Por que Assembly?

- Acesso direto ao hardware
  - Sistemas de software que requerem acesso direto ao hardware
    - Drivers; Interfaces de rede e outros dispositivos
- Espaço
  - Embora atualmente espaço não seja um diferencial para a maioria das aplicações, código compacto ainda pode ser importante em algumas situações:
    - Software para dispositivos portáteis em geral

Por que Assembly?

- Outras Aplicações:
  - Engenharia Reversa
  - Vírus e Anti-Vírus
  - Boot-Loaders
  - ...

Revisão

- Conceitos e Relacionamento:
  - Linguagens de Alto Nível
  - Assembly
  - Linguagem de Máquina
- Motivação:
  - Por que aprender Assembly?
  - Processo de “Compilação”
  - Eficiência e acesso direto ao hardware

Exercícios

1. Explique o relacionamento entre:

   a. Linguagem Assembly e Linguagem de Máquina

   b. Linguagem Assembly e Linguagem de Alto Nível

   Há relacionamento um-para-um entre as instruções?

   Em que casos?

2. Discuta explique exemplos de utilização de Assembly.

Hello World

```
section .text
    global _start                       ;must be declared for linker (ld)

_start:                                 ;tell linker entry point

        mov     edx,len ;message length
        mov     ecx,msg ;message to write
        mov     ebx,1   ;file descriptor (stdout)
        mov     eax,4   ;system call number (sys_write)
        int     0x80    ;call kernel

        mov     eax,1   ;system call number (sys_exit)
        int     0x80    ;call kernel

section .data

msg     db      'Hello, world!',0xa     ;our dear string
len     equ     $ - msg                 ;length of our dear string
```

Bibliografia

- Dandamudi, S. *Introduction do Assembly Language Programming: For Pentium and RISC Processors*. Springer, 2005.