

# Execução detalhada de instruções

## TPC4 + Guião Teatral

Alberto José Proença

---

### Objectivos

**Treinar as capacidades de visualização** de terminologia e conceitos que descrevem o funcionamento de um sistema de computação na execução de código.

Para atingir estes objetivos vai-se realizar um exercício experimental com estudantes-atores, em *contra-relógio*: usando 8 atores: "banco de registos", "ALU", "unidade de controlo do processador", "descodificador de instruções", "memória", "barramento de endereços", "barramento de dados", e "barramento de controlo". Estes irão representar (teatralmente) a execução de um conjunto de instruções em linguagem máquina, o corpo de uma função em C compilada e montada para uma arquitetura IA-16.

Como preparação para esta peça de teatro, são propostos alguns exercícios sob a forma de TPC. Estes deverão ser resolvidos e entregues no início da próxima sessão PL, para discussão sucinta no início dessa sessão, seguindo-se a peça de teatro propriamente dita (esta irá durar 2 semanas consecutivas).

---

### 1. Exercícios de preparação (TPC)

1. Considere a execução duma operação aritmética "montada" em linguagem máquina para a arquitetura IA-16 definida neste TPC, desde que o processador terminou a instrução anterior; em *assembly* corresponde a `addw %bx, -8(%bp)`.

Essa instrução dá indicação ao processador para adicionar 2 operandos de 16 bits – 1 colocado em registo e outro em memória – e guardar o resultado de volta nas mesmas células de memória onde antes estava guardado o 2º operando. O 1º operando está no registo `%bx`, enquanto o 2º operando está localizado em memória a partir do endereço calculado pela soma do conteúdo do registo `%bp` com a constante `(-8)`. Considere os valores em memória e nos registos apresentados neste enunciado nas pág. 4 e 5, e que esta instrução está codificada em 2 bytes na memória, referenciados pelo IP (veremos depois que isto é falso).

**Indique**, cronologicamente e em binário ou hexadecimal, toda a informação que irá circular nos 3 barramentos (ver sua descrição no exercício 2.) durante a execução integral desta instrução (não esquecer que o processador tem de ir buscar a instrução à memória). Considere que o barramento de dados transporta a informação de/para a memória sob a forma *little endian*, i.e., o byte menos significativo do barramento refere-se ao conteúdo da célula de memória com o endereço mais baixo.

**Indique** também todos os registos e todas as células de memória que foram modificados com a execução desta instrução.

2. Considere a operação de montagem em binário dessa mesma instrução em *assembly* de acordo com as regras definidas neste enunciado (inclui pág. 3): `addw %bx, -8(%bp)`.

**Tente** construir a instrução em linguagem máquina deste sistema IA-16 (em hexadecimal, *byte a byte*), depois de montada pelo *assembler*. **Explicita** sucintamente, o processo de montagem.

### 2. Caracterização da "peça de teatro"

Pretende-se com este exercício analisar todos os passos da execução de instruções por um processador *little endian* de 16 bits (semelhante ao Intel x86), desde a busca de cada uma das instruções à memória até à sua execução, passando pela sua descodificação e atualização do apontador para a próxima instrução, IP.

Os principais componentes do computador serão representados por estudantes-atores, estando cada uma/um apenas na posse da informação que necessita e durante o tempo que essa informação existe.

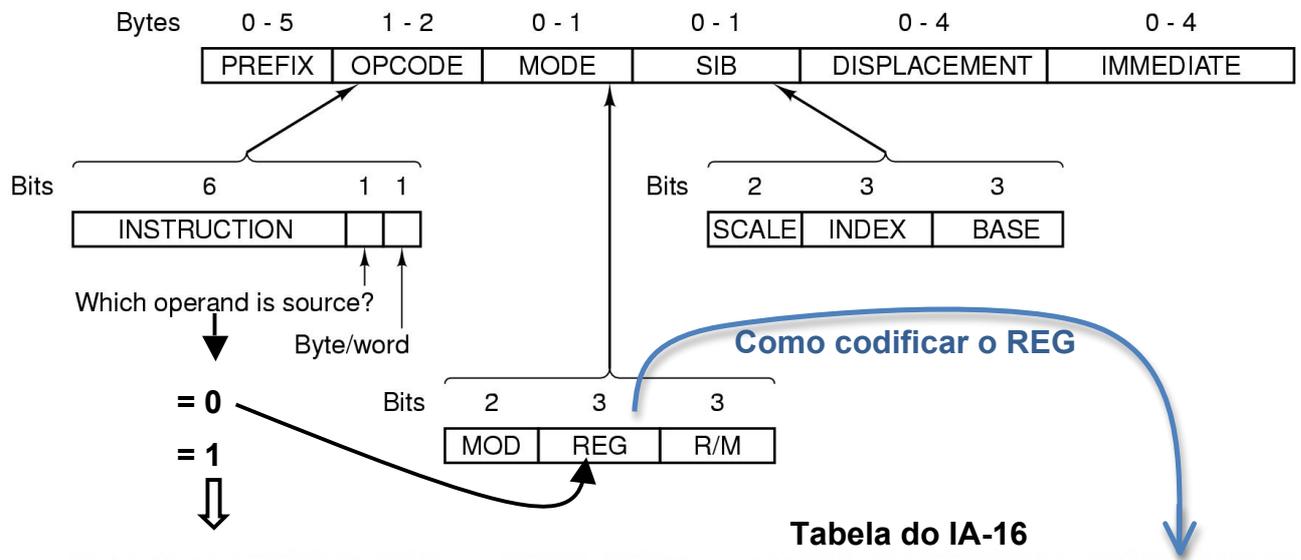
**Caraterísticas do sistema de computação e funções a desempenhar por cada estudante-ator:**

1. **Processador**, constituído pelas seguintes partes/atores:
  - a. **Banco de registos**, responsável pelo conteúdo dos 8 registos "genéricos" do Intel x86 (*ax*, *bx*, *cx*, *dx*, *si*, *di*, *bp*, *sp*) e do *instruction pointer* (*ip*); no início do exercício, os registos terão um conteúdo pré-definido (folha com o ator; contém a lista de registos e respectivo conteúdo inicial, bem como espaço para escrever os novos valores dos registos);
  - b. **ALU**, responsável por efetuar as operações aritméticas (soma/subtração) ou lógicas (AND/OR/NOT) que lhe forem solicitadas, e sobre os operandos que lhe forem disponibilizados; no fim o resultado necessita de ser armazenado algures; as operações são feitas no quadro e apagadas após a sua conclusão (a ALU não tem capacidade de armazenar valores);
  - c. **Unidade de controlo**, responsável por gerar todos os sinais que controlam as operações no exterior do processador, e ainda por dar todas as instruções para o correto funcionamento interno do processador; a apoiá-la/o terá a colaboração de uma outra estrutura/ator (o decodificador de instruções);
  - d. **Decodificador de instruções**, com capacidade para armazenar internamente até 4 *bytes* com instruções em binário; a decodificação das instruções faz-se com base na informação disponibilizada na pág. 3, contendo:
    - (i) figura com os formatos de instruções do i386,
    - (ii) mapa da codificação dos modos de endereçamento do i386, em que a última coluna mostra também como os registos são codificados, e
    - (iii) tabela com códigos de operação das instruções mais usadas nesta peça; de notar que este mapa dos modos de endereçamento se refere a um processador de 32 bits, mas que iremos adaptá-lo nesta peça a um processador de 16 bits, com as necessárias correções (por ex., todas as referências a registos de 32 bits deverão ser substituídas por referências a registos de 16 bits);
2. **Memória**, responsável pelo conteúdo das  $2^{16}$  células de memória (folha com o ator; contém o conteúdo de células numa lista de endereços previamente definidos, bem como espaço para escrever novos valores em células que tenham sido modificadas);
3. **Barramentos** de interligação entre o processador e a memória:
  - a. **Barramento de endereços**, responsável por transportar 16 bits de cada vez (em 2 folhas de papel, 1 em cada mão contendo um valor numérico de 1 *byte*), e apenas durante o período de tempo em que esses valores estiverem ativos no barramento;
  - b. **Barramento de dados**, responsável por transportar 16 bits de cada vez (em 2 folhas de papel, 1 em cada mão contendo um valor numérico de 1 *byte*), e apenas durante o período de tempo em que esses valores estiverem ativos no barramento;
  - c. **Barramento de controlo**, responsável por transportar os sinais de controlo que forem necessários (neste exercício apenas serão necessários os sinais de RD e WR).

**3. Guião teatral**

1. Distribuir os papéis com a informação pertinente a cada um dos atores, conjuntamente com várias folhas de papel para que a "PU" e a "memória" possam escrever a informação que os "barramentos" irão transportar.
2. Considerar que a PU acabou de executar uma instrução, e que o estado do computador é o que está representado nas folhas distribuídas.
3. Simular com as/os atores a execução de instruções até ao fim da 1ª instrução de `ret` que encontrar.
4. (Para fazer depois da aula) Tentar recriar o código em C que deu origem a esta função compilada. (Sugestão: dê uma vista de olhos pelos slides das aulas...)

# Formatos de instrução no IA-32



Operands	Memory Operands			Register Operands	
	No Displacement	Displacement 8-bit	Displacement 16-bit	W = 0	W = 1
MOD	00	01	10	11	
R/M				W = 0	W = 1
000	(BX) + (SI)	(BX) + (SI) + D8	(BX) + (SI) + D16	AL	AX
001	(BX) + (DI)	(BX) + (DI) + D8	(BX) + (DI) + D16	CL	CX
010	(BP) + (SI)	(BP) + (SI) + D8	(BP) + (SI) + D16	DL	DX
011	(BP) + (DI)	(BP) + (DI) + D8		BL	BX
100	(SI)	(SI) + D8	(SI) + D16	AH	SP
101	(DI)	(DI) + D8	(DI) + D16	CH	BP
110	D16	(BP) + D8	(BP) + D16	DH	SI
111	(BX)	(BX) + D8	(BX) + D16	BH	DI

Opcode	Mnemónica	Comentários
0000 00xx	add	xx: ver figura acima; requer mais bytes
0101 0yyy	push	yyy: identificação de registo, de acordo com tabela acima
0101 1yyy	pop	yyy: identificação de registo, de acordo com tabela acima
1000 10xx	mov	xx: ver figura acima; requer mais bytes
1000 110x	lea	xx: ver figura acima; requer mais bytes
1100 0011	ret	

## Banco de registos

<b>ax</b>	xxxx xxxx xxxx xxxx	0 0 0 c	_____	_____
<b>bx</b>	xxxx xxxx xxxx xxxx	f f e f	_____	_____
<b>cx</b>	xxxx xxxx xxxx xxxx	0 0 0 0	_____	_____
<b>dx</b>	xxxx xxxx xxxx xxxx	0 1 4 0	_____	_____
<b>si</b>	xxxx xxxx xxxx xxxx	8 0 8 0	_____	_____
<b>di</b>	xxxx xxxx xxxx xxxx	8 0 c 6	_____	_____
<b>bp</b>	xxxx xxxx xxxx xxxx	8 4 1 c	_____	_____
<b>sp</b>	xxxx xxxx xxxx xxxx	8 4 1 4	_____	_____
<b>ip</b>	xxxx xxxx xxxx xxxx	4 0 4 0	_____	_____
			_____	_____
			_____	_____
			_____	_____

# Memória

<b>0x0000</b>	0101 1001	<u>5</u> <u>9</u>	___	___	___	___	___
...	...						
<b>0x4040</b>	0101 0101	<u>5</u> <u>5</u>	___	___	___	___	___
<b>1</b>	1000 1001	<u>8</u> <u>9</u>	___	___	___	___	___
<b>0x4042</b>	1110 0101	<u>e</u> <u>5</u>	___	___	___	___	___
<b>3</b>	1000 1011	<u>8</u> <u>b</u>	___	___	___	___	___
<b>0x4044</b>	0100 0110	<u>4</u> <u>6</u>	___	___	___	___	___
<b>5</b>	0000 0110	<u>0</u> <u>6</u>	___	___	___	___	___
<b>0x4046</b>	0000 0011	<u>0</u> <u>3</u>	___	___	___	___	___
<b>7</b>	0100 0110	<u>4</u> <u>6</u>	___	___	___	___	___
<b>0x4048</b>	0000 0100	<u>0</u> <u>4</u>	___	___	___	___	___
<b>9</b>	1000 1001	<u>8</u> <u>9</u>	___	___	___	___	___
<b>0x404a</b>	1110 1100	<u>e</u> <u>c</u>	___	___	___	___	___
<b>b</b>	0101 1101	<u>5</u> <u>d</u>	___	___	___	___	___
<b>0x404c</b>	1100 0011	<u>c</u> <u>3</u>	___	___	___	___	___
<b>d</b>	1000 1101	<u>8</u> <u>d</u>	___	___	___	___	___
<b>0x404e</b>	0111 0110	<u>7</u> <u>6</u>	___	___	___	___	___
<b>f</b>	0000 0000	<u>0</u> <u>0</u>	___	___	___	___	___
...	...						
<b>0x8410</b>	0010 1001	<u>2</u> <u>9</u>	___	___	___	___	___
<b>1</b>	0001 1111	<u>1</u> <u>f</u>	___	___	___	___	___
<b>0x8412</b>	1101 0101	<u>d</u> <u>5</u>	___	___	___	___	___
<b>3</b>	0010 1001	<u>2</u> <u>9</u>	___	___	___	___	___
<b>0x8414</b>	0001 0010	<u>1</u> <u>2</u>	___	___	___	___	___
<b>5</b>	0100 0000	<u>4</u> <u>0</u>	___	___	___	___	___
<b>0x8416</b>	0001 0100	<u>1</u> <u>4</u>	___	___	___	___	___
<b>7</b>	0000 0000	<u>0</u> <u>0</u>	___	___	___	___	___
<b>0x8418</b>	1110 1010	<u>e</u> <u>a</u>	___	___	___	___	___
<b>9</b>	1111 1111	<u>f</u> <u>f</u>	___	___	___	___	___
...	...						

<b>Nº</b>	<b>Nome:</b>	<b>Turma:</b>
-----------	--------------	---------------

### Resolução dos exercícios

(**Nota:** Apresente sempre os cálculos que efectuar no verso da folha; o não cumprimento desta regra equivale à não entrega do trabalho.)

1. **Indique**, cronologicamente e em bin ou hex, toda a informação que irá circular nos 3 barramentos:

```
addw %bx, -8(%bp)
```

*Address Bus:*

*Data Bus:*

*Control Bus*  
(indique só os  
sinais de controlo):

**Indique** também todos os registos e todas as células de memória modificados: