

# Execução detalhada de instruções

## TPC3 + 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 actores - "banco de registos", "ALU", "unidade de controlo do CPU", "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 IA16).

Como preparação para esta peça de teatro, são propostos alguns exercícios sob a forma de TPC, os quais deverão ser resolvidos e entregues no início da sessão TP, para discussão durante esta sessão. A peça de teatro será na sessão TP seguinte.

---

### 1. Caraterização do problema

Pretende-se com este exercício analisar todos os passos da execução de instruções por um processador (um CPU semelhante ao Intel x86, também *little endian*), desde a busca de cada uma das instruções à memória, até à sua execução, passando pela sua descodificação.

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

#### Caraterísticas do sistema de computação e funções a desempenhar por cada estudante-actor:

- CPU, constituído pelas seguintes partes/atores:
  - **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);
  - **ALU**, responsável por efectuar 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);
  - **unidade de controlo**, responsável por gerar todos os sinais que controlam as operações no exterior do CPU, e ainda por dar todas as instruções para o correto funcionamento interno do CPU; a apoiá-la/o terá a colaboração de uma outra estrutura/ator (o descodificador de instruções);
  - **descodificador de instruções**, com capacidade para armazenar internamente até 4 *bytes* com instruções em binário; a descodificação 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 correcções (por ex., todas as referências a registos de 32 bits deverão ser substituídas por referências a registos de 16 bits);

- **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);
- barramentos de interligação entre o CPU e a memória:
  - **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;
  - **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;
  - **barramento de controlo**, responsável por transportar os sinais de controlo que forem necessários (neste exercício apenas serão necessários os sinais de RD e WR).

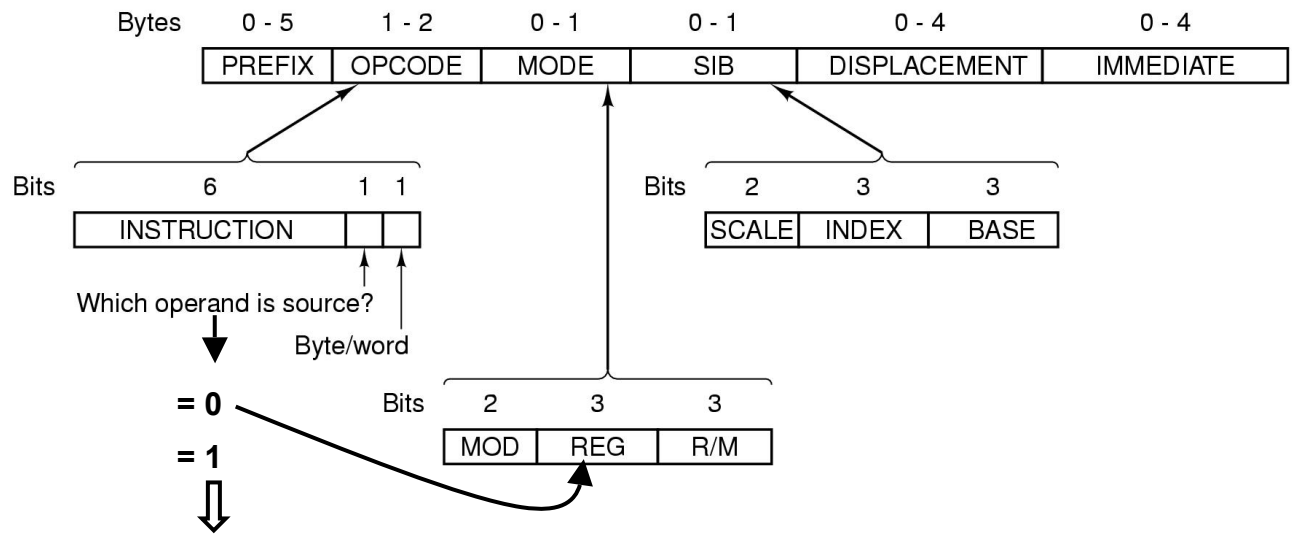
## 2. Exercícios de preparação (TPC)

1. Considere a execução da instrução em *assembly* `addw %bx, -8(%bp)`, (depois de "montada" em linguagem máquina para a arquitetura IA16 definida neste TPC) desde que o CPU terminou a instrução anterior. Essa instrução instrui o CPU para adicionar 2 operandos de 16 bits – 1 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)`.  
**Indique**, cronologicamente e em binário ou hexadecimal, toda a informação que irá circular nos 3 barramentos durante a execução integral desta instrução (não esquecer que o CPU tem de ir buscar a instrução à memória). Considere os valores apresentados neste enunciado.  
**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: `addw %bx, -8(%bp)`.  
**Mostre** como seria esta instrução em linguagem máquina deste IA16 (em hexadecimal, *byte a byte*), depois de montada pelo *assembler*. **Explique** sucintamente, o processo de montagem.

## 3. Guião teatral

1. Distribuir os papéis com a informação pertinente a cada um dos atores, conjuntamente com várias folhas de papel para que o "CPU" e a "memória" possam escrever a informação que os "barramentos" irão transportar.
2. Considerar que o CPU 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 acetatos das aulas...*)

# Formatos de instrução



MOD				
R/M	00	01	10	11
000	M[EAX]	M[EAX + OFFSET8]	M[EAX + OFFSET32]	EAX or AL
001	M[ECX]	M[ECX + OFFSET8]	M[ECX + OFFSET32]	ECX or CL
010	M[EDX]	M[EDX + OFFSET8]	M[EDX + OFFSET32]	EDX or DL
011	M[EBX]	M[EBX + OFFSET8]	M[EBX + OFFSET32]	EBX or BL
100	SIB	SIB with OFFSET8	SIB with OFFSET32	ESP or AH
101	Direct	M[EBP + OFFSET8]	M[EBP + OFFSET32]	EBP or CH
110	M[ESI]	M[ESI + OFFSET8]	M[ESI + OFFSET32]	ESI or DH
111	M[EDI]	M[EDI + OFFSET8]	M[EDI + OFFSET32]	EDI or BH

Opcode	Mnemónica	Comentários
0000 00xx	add	xx: ver figura acima; requer mais <i>bytes</i>
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 <i>bytes</i>
1000 110x	lea	xx: ver figura acima; requer mais <i>bytes</i>
1100 0011	ret	

## Banco de registos

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

# 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 0101	<u>4</u> <u>5</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 0101	<u>4</u> <u>5</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 apenas os sinais de controlo):

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

2. **Mostre** como seria esta instrução em linguagem máquina deste IA16 (**explique** como lá chegou)