

# Execução detalhada de instruções

## TPC3 + Guião

Baseado no guião de Alberto José Proença

---

### Metodologia

Leia o enunciado e responda manualmente às questões na folha fornecida para o efeito.

A resolução deverá ser entregue **impreterivelmente** no início da sessão PL, com a presença do estudante durante a sessão PL para que o TPC seja contabilizado na avaliação por participação. Não serão aceites trabalhos entregues fora da PL.

### Objetivos

**Treinar as capacidades de visualização** do funcionamento de um sistema de computação na execução de código.

Para atingir estes objetivos vai-se realizar um exercício simulado sob a forma de uma peça teatral, usando 8 estudantes-atores: "banco de registos", "ALU", "unidade de controlo", "descodificador de instruções", "memória" e os 3 barramentos, "Address Bus", "Data Bus" e "Control Bus".

Os estudantes irão simular a execução de um conjunto de instruções em linguagem máquina, que corresponde à execução do corpo de uma função em C compilada e montada para uma arquitetura IA-16.

Como preparação para esta aula são propostos exercícios sob a forma de TPC, para discussão no início da sessão PL, seguindo-se a simulação propriamente dita (esta irá necessitar de mais uma sessão PL).

---

## 1. Exercícios de preparação (TPC)

Considere a execução de uma operação aritmética em linguagem máquina da arquitetura IA-16 definida neste TPC, desde que o processador terminou a instrução anterior; em *assembly* essa operação corresponde a **addw -8(%bp), %ax**<sup>1</sup>.

Considere ainda que: (i) os valores em memória e nos registos são os que estão indicados na figura no topo da página 3, (ii) esta instrução está codificada em 2 bytes na memória; (veremos depois que esta informação não está 100% correta).

- Indique**, cronologicamente (em binário ou hexadecimal), toda a informação que irá circular nos 3 barramentos (ver a descrição do sistema no ponto 2) durante a execução desta instrução (não esquecer que a unidade de processamento 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 que foram modificados com a execução desta instrução.
- Considere o processo de codificação das operações de acordo com o formato de instrução IA-16 indicado na figura da página 3.

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

---

<sup>1</sup> Esta instrução dá indicação ao processador para adicionar 2 operandos de 16 bits – 1 colocado em registo e outro em memória – e guardar o resultado no registo %ax. O 1º operando está no registo %ax enquanto o 2º operando está localizado em memória a partir do endereço calculado pela soma do conteúdo do registo %bp com a constante (-8).

## 2. Caracterização do sistema de computação

Pretende-se analisar todos os passos da execução de instruções por uma unidade de processamento *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.

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

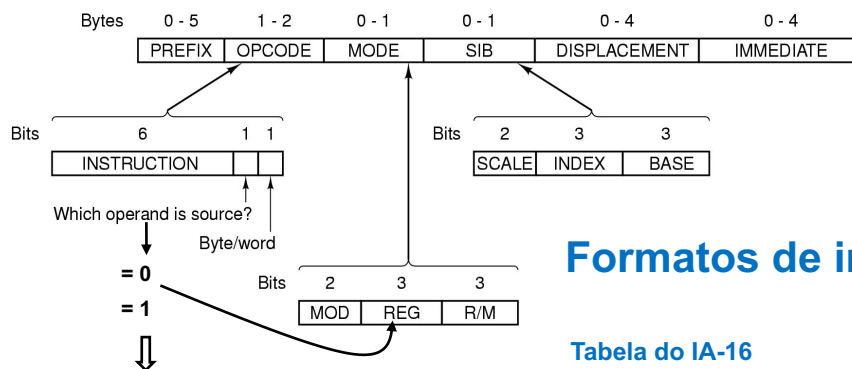
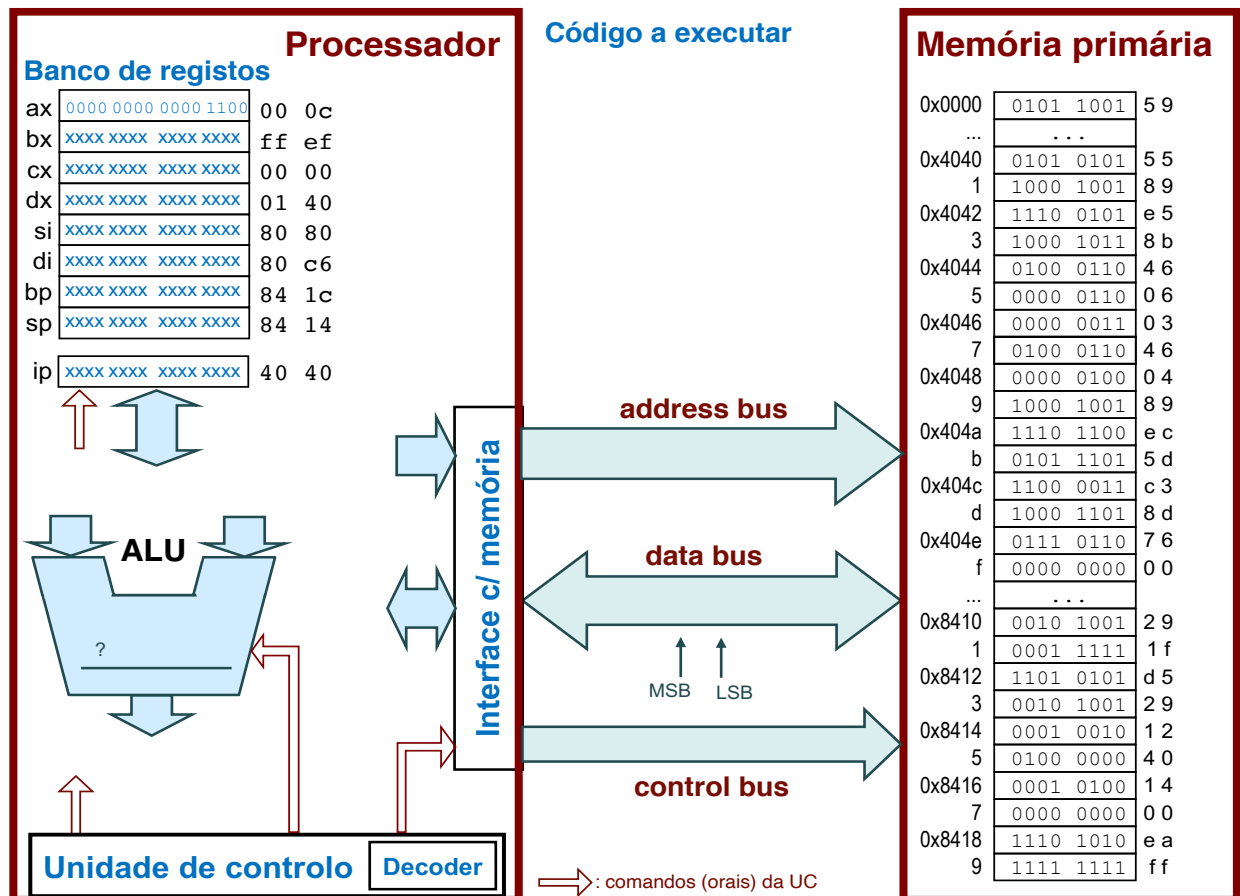
1. **Unidade de processamento**, constituída pelas seguintes partes/atores:
  - a. **Unidade de controlo**, responsável por dar, no *timing* apropriado, todas as indicações para o correto funcionamento do sistema, e ainda por gerar os sinais que controlam as operações no exterior da unidade de processamento. Esta unidade inclui um descodificador de instruções:  
**Descodificador de instruções**, com capacidade para armazenar internamente vários *bytes* com instruções em binário; a descodificação das instruções faz-se com base na informação disponibilizada na figura/tabelas da página 3;
  - b. **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*); a figura contém a lista de registos e respectivo conteúdo inicial;
  - c. **ALU**, responsável por efetuar as operações sobre os operandos que lhe forem disponibilizados; no fim o resultado necessita de ser armazenado algures;
2. **Memória principal**, responsável pelo conteúdo das  $2^{16}$  células de memória; a figura contém o conteúdo de células numa lista de endereços previamente definidos.

Adicionalmente o sistema dispõe de **barramentos** de interligação entre o processador e a memória:

- a. **Barramento de endereços**, responsável por transportar 16 bits de cada vez, organizados em duas partes de 1 *byte* cada: a de maior valor MSB (*most significant byte*) e a de menor valor, LSB (*least significant byte*);
- b. **Barramento de dados**, responsável por transportar 16 bits de cada vez, organizados também em duas partes de 1 *byte* cada;
- 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 para a peça teatral

1. Selecionar equipas de 8 atores para execução de cada uma das instruções e atribuir os papéis a cada elemento da equipa; um dos atores irá registando no quadro as ações e respetiva evolução da execução.
2. Considerar que o estado inicial do computador é o representado nas figuras e que este irá iniciar a execução de uma nova instrução.
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...)



Operands	Memory Operands			Register Operands	
	No Displacement	Displacement 8-bit	Displacement 16-bit	11	
	MOD 00	01	10	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 reg, de acordo com tabela acima
0101 1yyy	pop	yyy: identificação de reg 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	

**Nº****Nome:****Turma:****Resolução dos exercícios**

(**Nota:** Apresente sempre os cálculos que efetuar; 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 -8(%bp), %ax
```

*Address Bus:*

*Data Bus:*

*Control Bus*

(indique só os  
sinais de controlo):

**Indique** também todos os registos modificados: