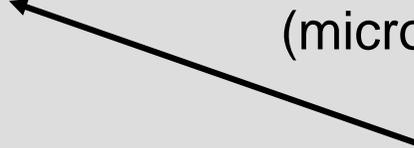


Arquitectura e Organização de Computadores



(micro-arquitectura)

atributos visíveis ao programador:

- I.S.A.
- tamanho da palavra (*bits*)
- registos

Componentes que realizam a arquitectura:

- organização do CPU (*pipeline*, ...)
- unidades específicas (FPU, MM, ...)
- barramentos (largura, velocidade)
- frequência do relógio

A existência de uma instrução de multiplicação é uma questão de arquitectura ou organizacional?

E a implementação desta instrução com *hardware* apropriado ou usando adições sucessivas?

Arquitectura e Organização de Computadores

Exemplo: IBM System/360

Lançado em 1965, incluía 4 modelos (40, 50, 65, 75) com a mesma arquitectura e diferentes organizações.

Preços mais baixos correspondiam a piores desempenhos.

Exemplo: Intel x86

Os processadores mais recentes, com uma organização cada vez mais complexa e melhores desempenhos, mantêm compatibilidade binária com os processadores anteriores.

Processadores da mesma geração têm organizações diferentes por razões de consumo (potência dissipada) e preço.

Arquitectura e Organização de Computadores

- Diferentes organizações motivadas por :
 - diferentes requisitos, e.g., desempenho, custo, potência dissipada;
 - avanços tecnológicos
- Alterações funcionais implicam alterações na arquitectura, pois têm que ser expostas aos programadores

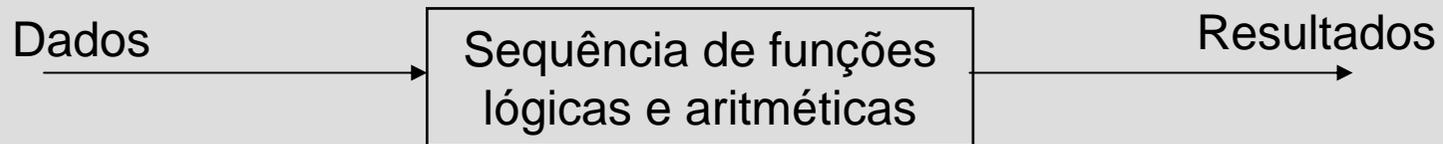
Máquina de von Neumann

John von Neumann, “First Draft of a Report on the EDVAC”,
Moore School of Electrical Engineering, Univ. of Pennsylvania
June, 30, 1945

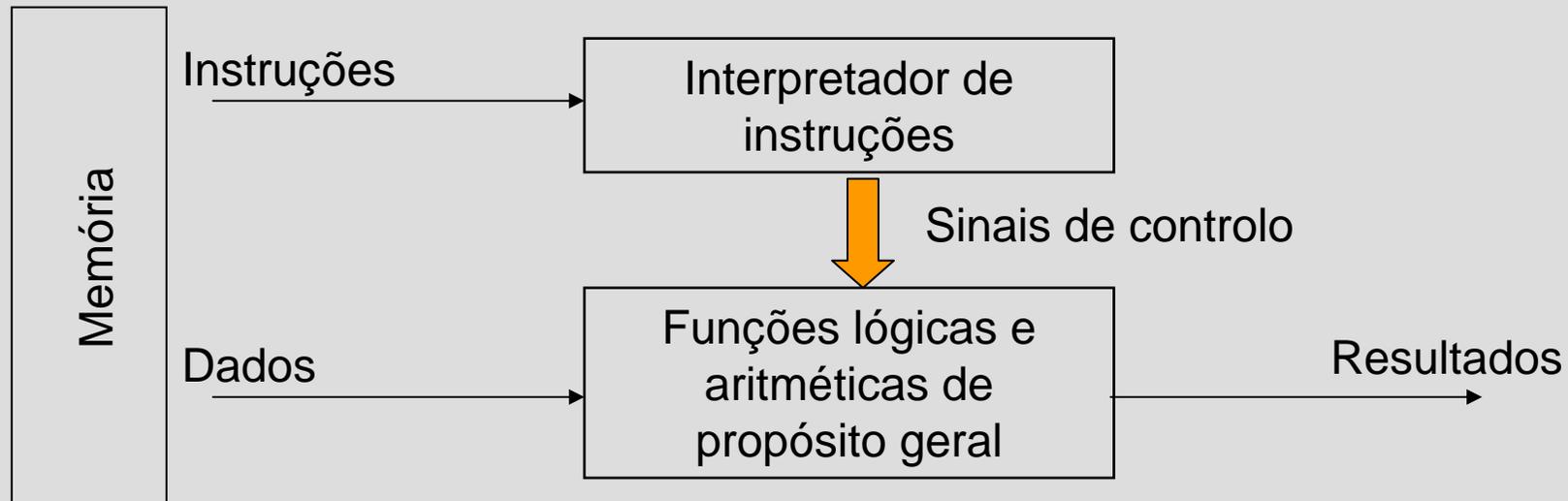
Três contribuições fundamentais:

1. Conceito dos programas armazenados
2. Organização básica de um computador
3. Arquitectura básica (tipos de instruções)

A arquitectura de von Neumann



a) Visão sistémica

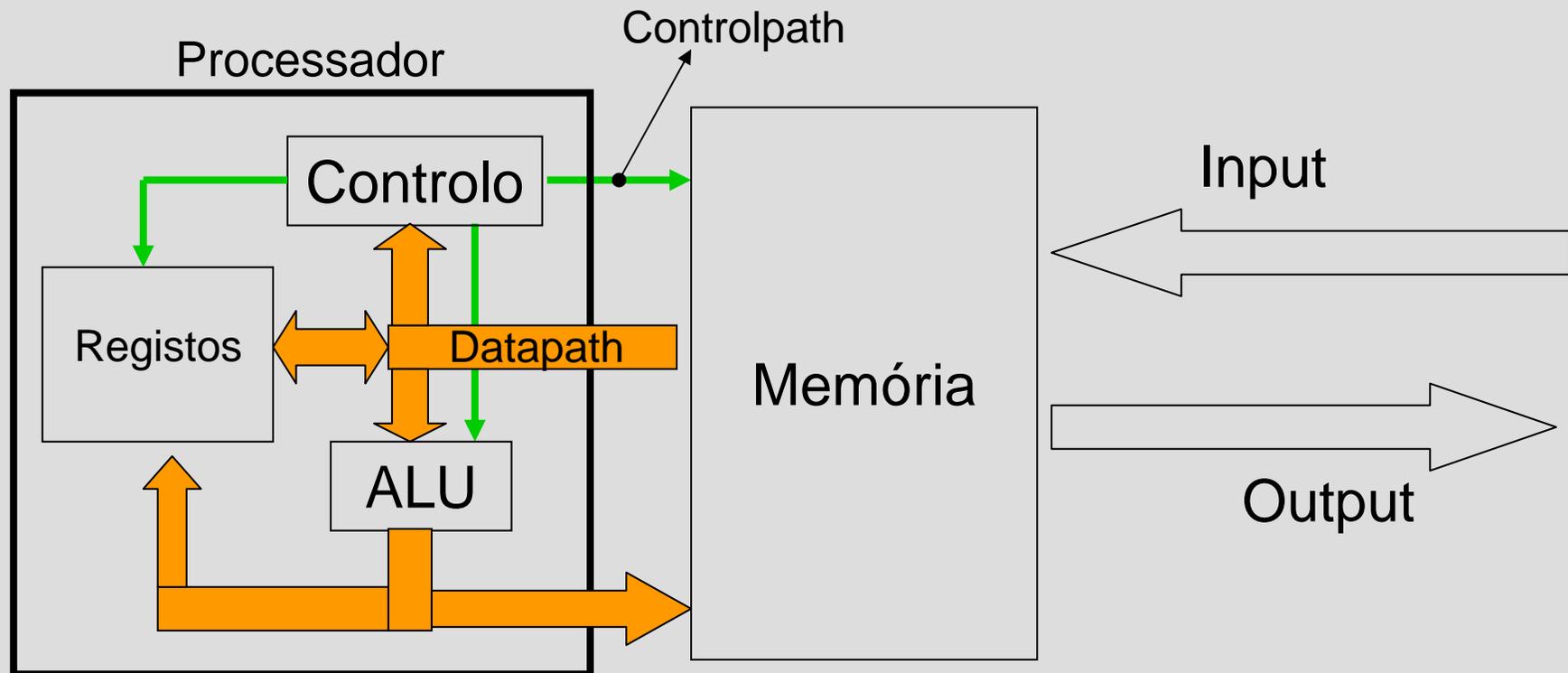


b) Visão estrutural

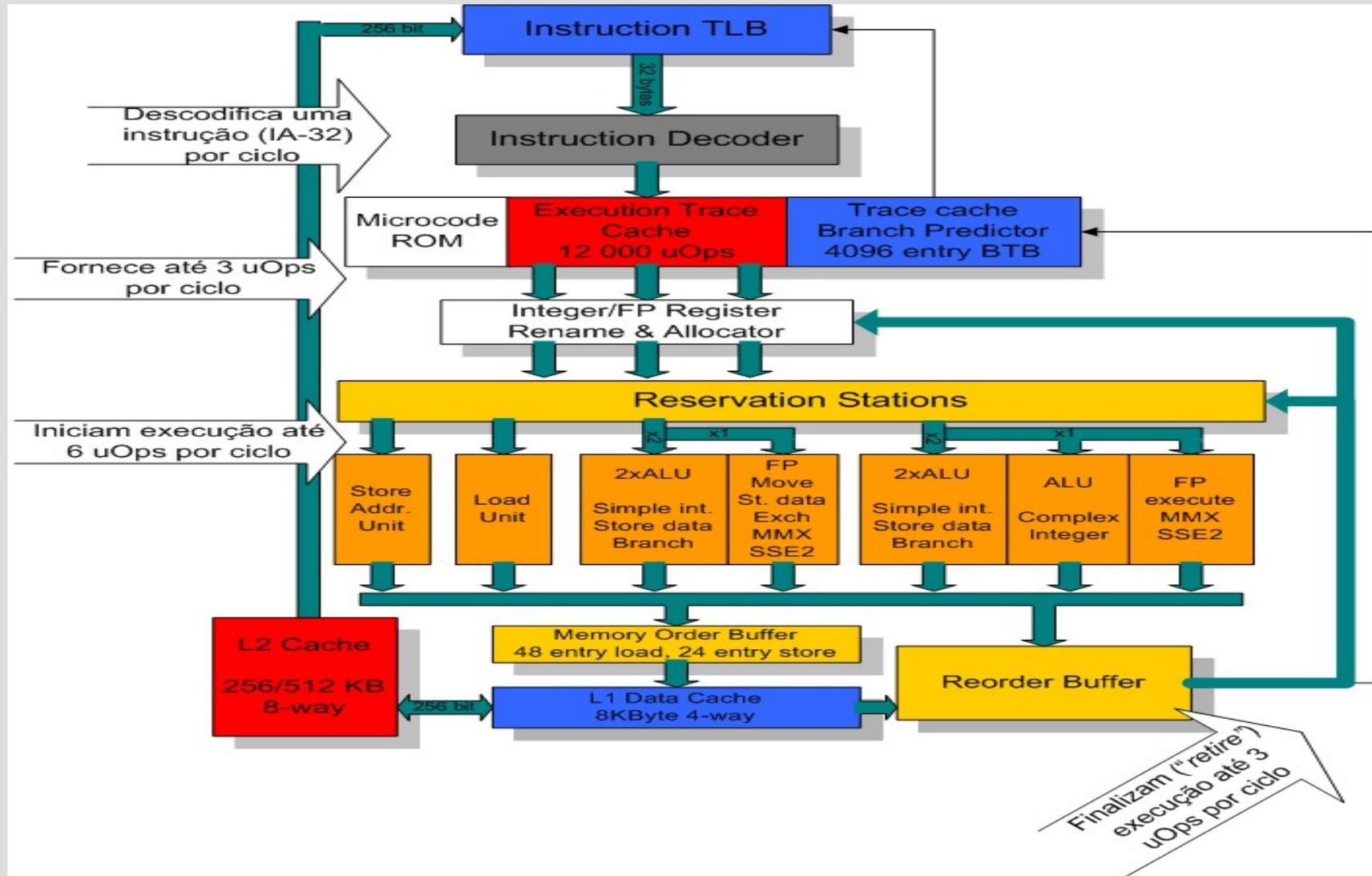
A arquitectura de von Neumann

- Conceito dos programas armazenados
 - O programa consiste em instruções binárias, que são executadas sequencialmente, e que estão armazenadas em posições consecutivas de memória
 - A unidade de controlo descodifica cada instrução e gera os sinais de controlo necessários para que os restantes componentes executem essa instrução
 - O computador pode ser reprogramado, alterando apenas o conteúdo da memória

A organização de von Neumann



Organização Intel Pentium IV



A arquitetura de von Neumann

(tipos de instruções)

“The really decisive considerations [...] in selecting an [instruction set] are more of a practical nature: simplicity of the equipment demanded[...] and the clarity of its application to the actually important problems[...].”

Burks, Goldstine and von Neumann, 1947

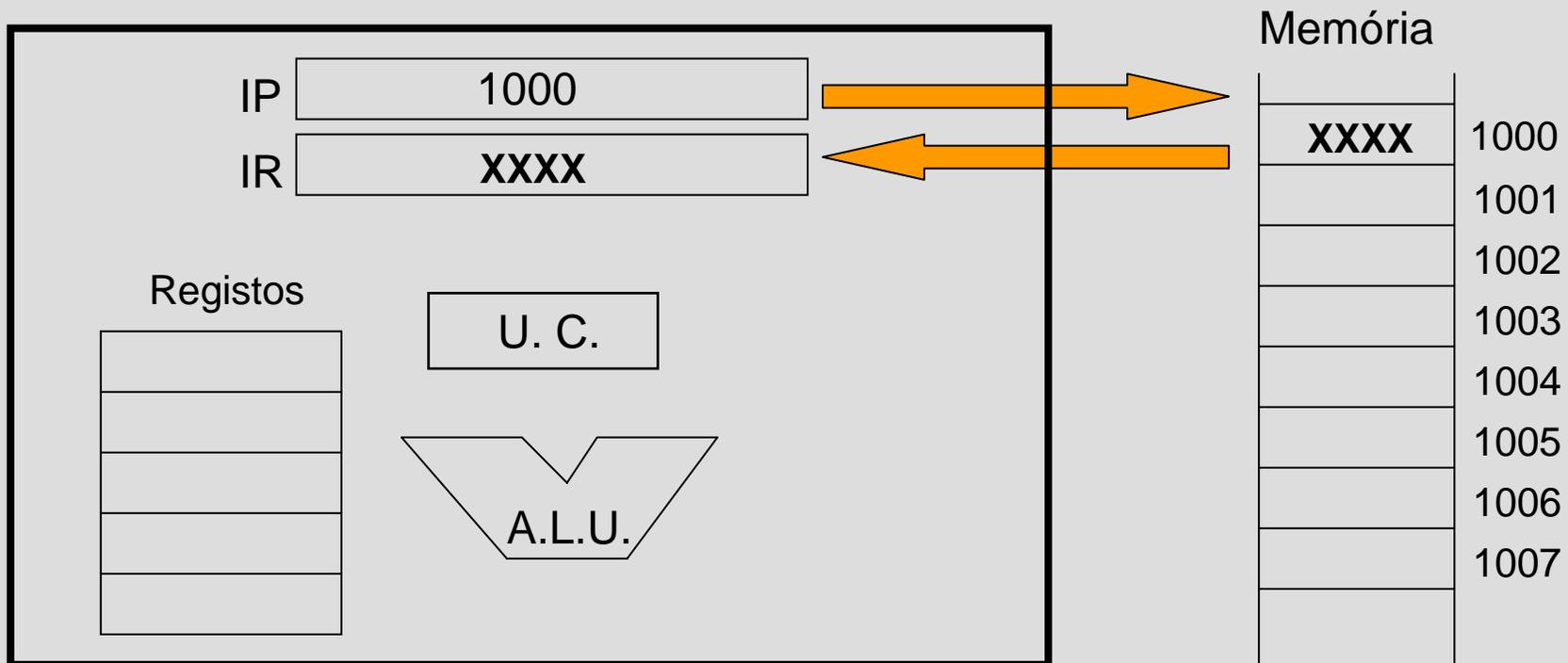
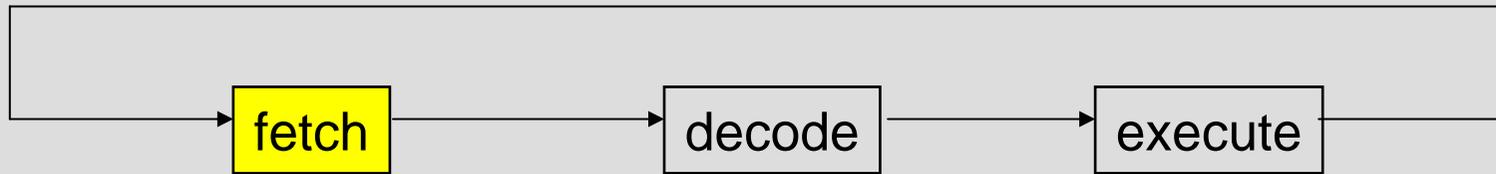
- Operações de cálculo. Ex.: add, sub, or, and, etc...
- Operações de transferência de dados:
 - reg ↔ reg, reg ↔ mem, mem ↔ mem, input/output

“The utility of a [...] computer lies in the possibility of using a given sequence of instructions repeatedly, the number of times it is iterated being dependent upon the results of the computation.”

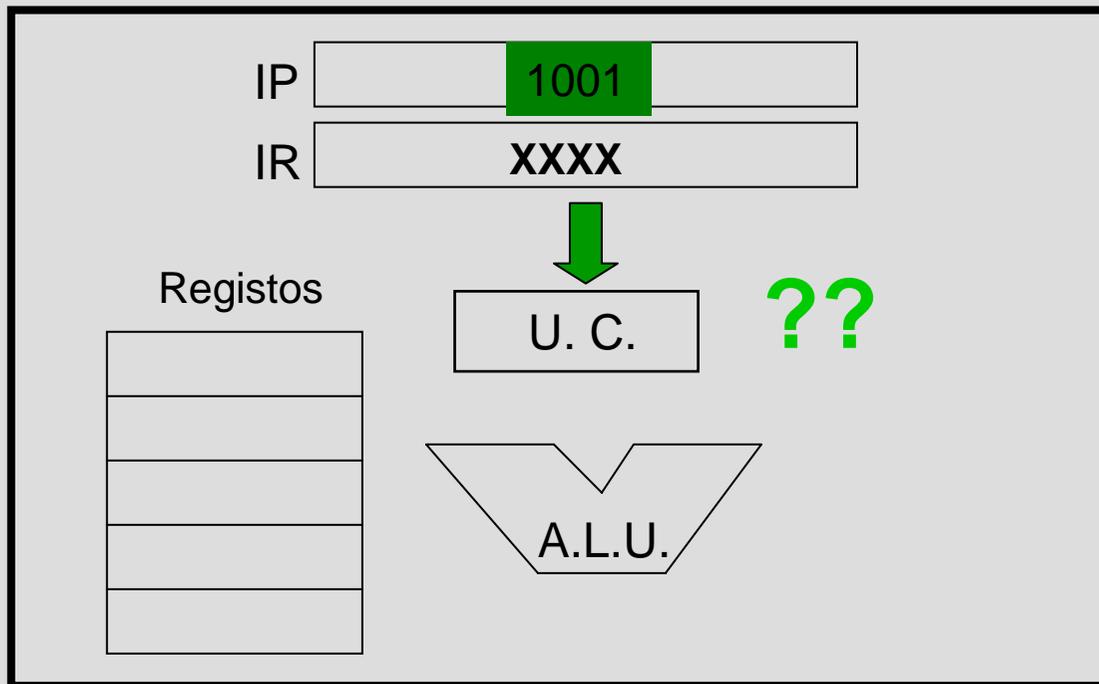
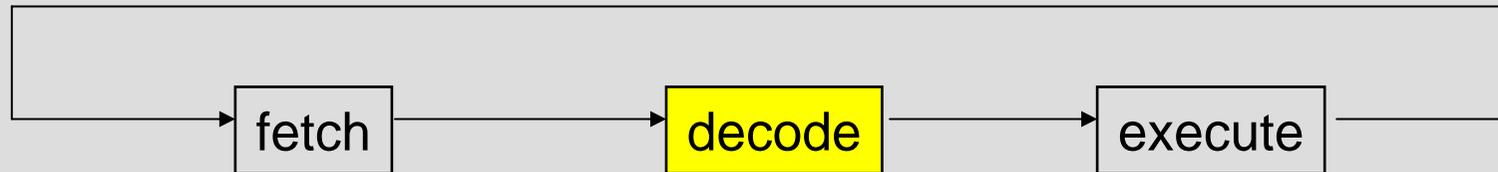
Burks, Goldstine and von Neumann, 1947

- Operações de controlo de fluxo condicional:
 - saltar se zero, saltar se diferente, saltar se maior, etc...

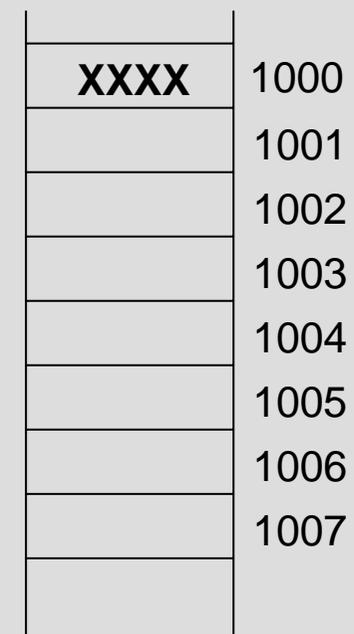
O ciclo do processador - *fetch*



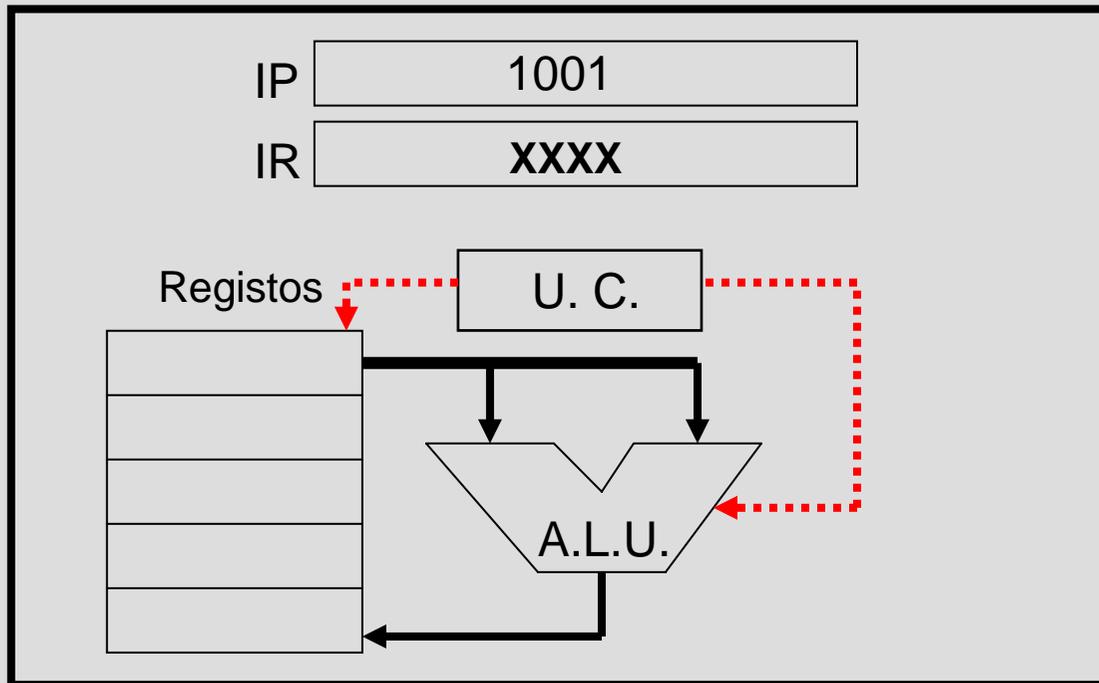
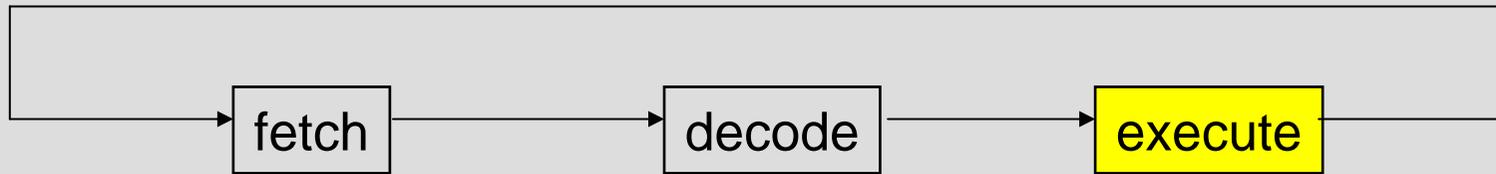
O ciclo do processador - *decode*



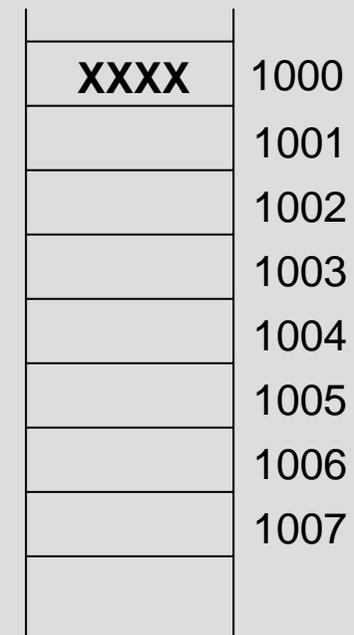
Memória



O ciclo do processador - *execute*



Memória

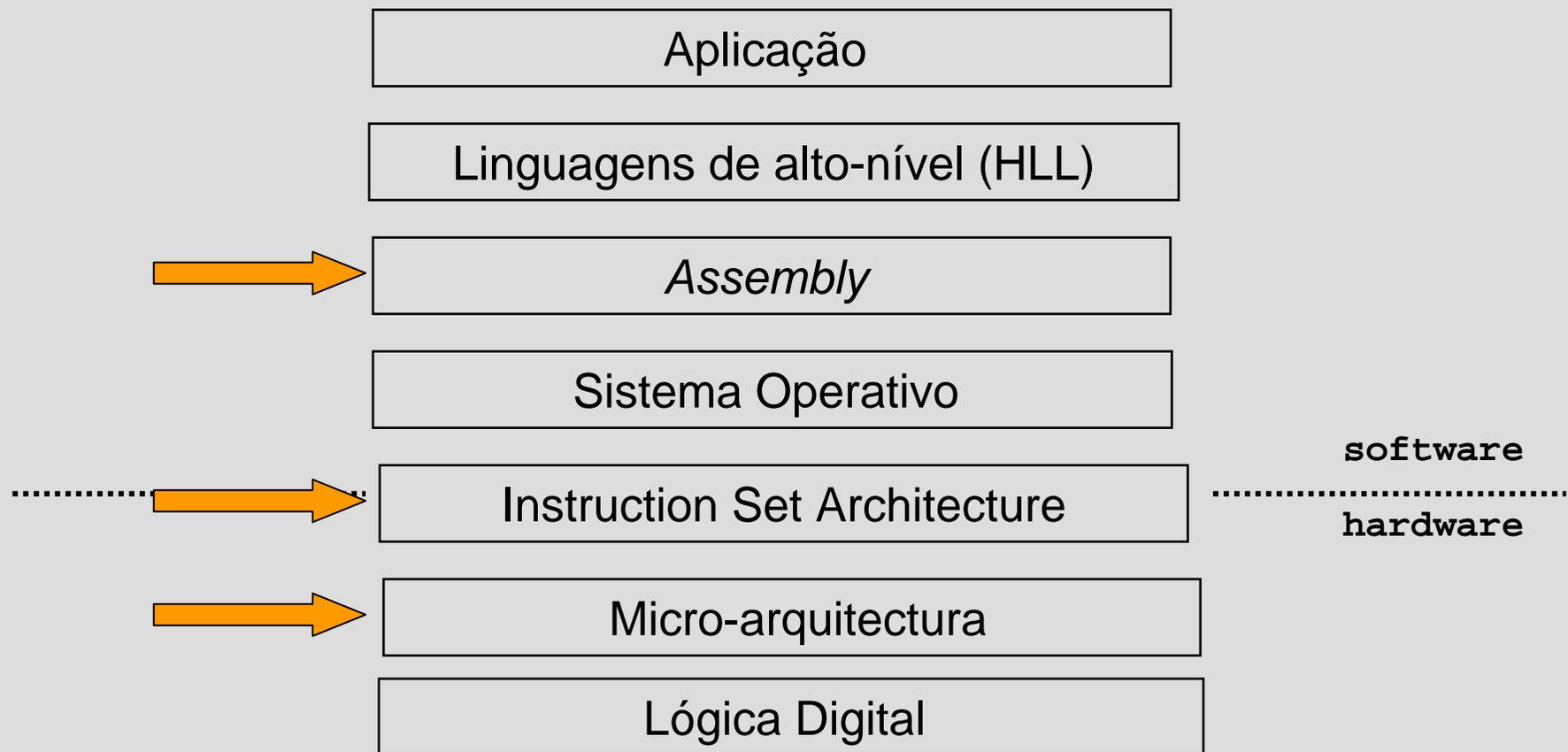


Níveis de abstracção

Computador como uma pilha de máquinas virtuais.

Cada nível usa os serviços disponibilizados pelo nível abaixo.

Cada utilizador usa o nível mais conveniente para a tarefa a resolver.



Níveis de abstracção

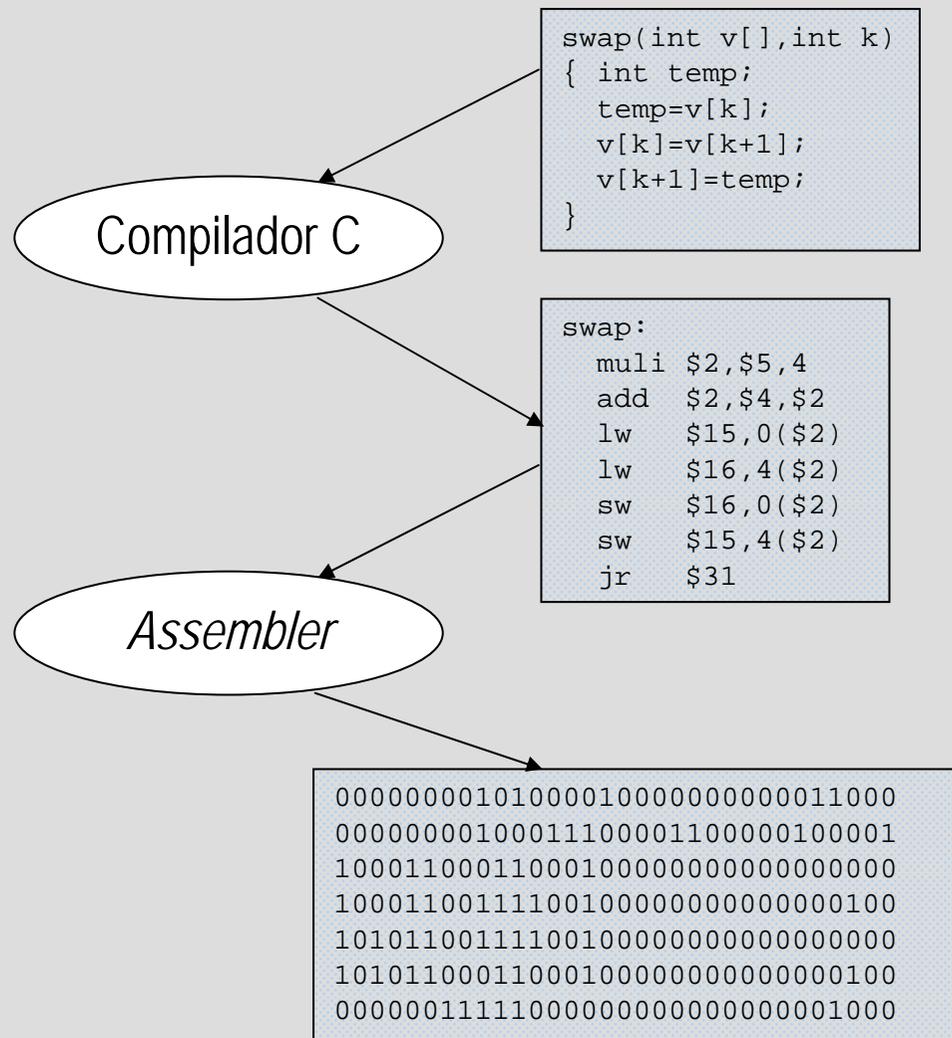
Compiladores – convertem um programa para um nível inferior

Interpretadores – executam instrução a instrução um programa de um nível superior

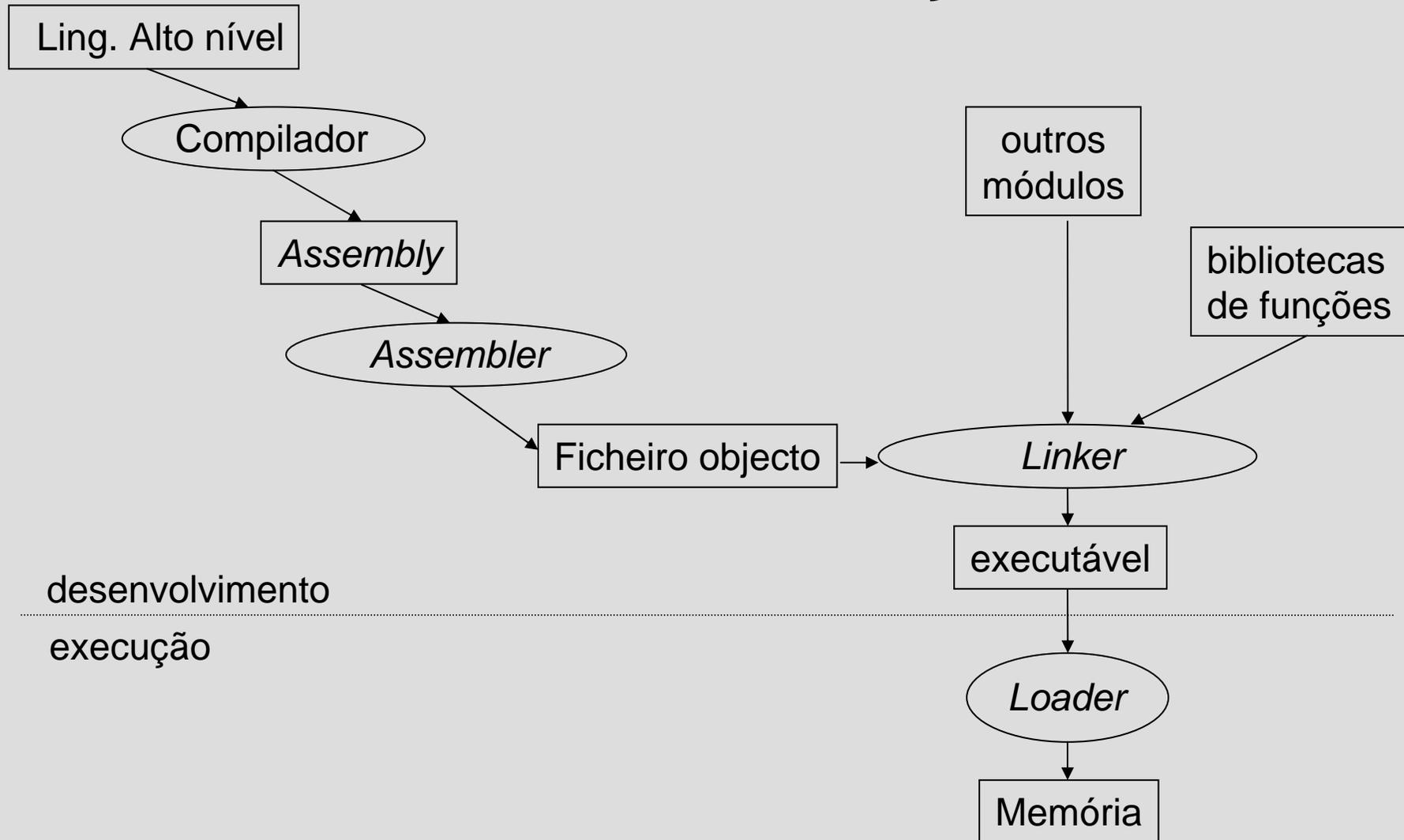
Assemblers – convertem um programa de assembly para o nível máquina

Linkers – ligam vários módulos de um mesmo programa, para gerar um único executável

Loaders – carregam um ficheiro executável para memória



Níveis de abstracção



Compilação

Módulo 1

```
int total=0;
main ()
{  int i;

    i = 10;
    soma (i);
}
```

```
.globl
total: .long 0
main:
    pushl %ebp
    movl  %esp, %ebp
    pushl $10
    call  soma
    leave
    ret
```

Módulo 2

```
extern int total;

void soma (int p)
{
    total += p;
}
```

```
.globl
soma:
    pushl %ebp
    movl  total, %eax
    movl  %esp, %ebp
    addl  8(%ebp), %eax
    movl  %eax, total
    leave
    ret
```

Compilação

Após a compilação o código *assembly* mantém informação simbólica.

Montagem (*Assembler*)

Ficheiros Objecto

```
00000000 <main>:
0: 55
1: 89 e5
c: 6a 0a
e: e8 fc ff ff ff
13: c9
14: c3
TABELA DE SÍMBOLOS
soma
```

```
00000000 <soma>:
0: 55
1: a1 00 00 00 00
6: 89 e5
8: 03 45 08
b: a3 00 00 00 00
10: c9
11: c3
TABELA DE SÍMBOLOS
total
```

O ficheiro objecto de cada um dos módulos não contém informação simbólica.

As instruções, representadas por mnemónicas após a compilação, são convertidas no código binário correspondente ao nível máquina..

Os endereços que não podem ser determinados não são preenchidos.

Os símbolos cujo endereço será determinado pelo *linker* são guardados na tabela de símbolos.

Linker

Ficheiro Executável

```
080482f4 <main>:
 80482f4: 55
 80482f5: 89 e5
 8048300: 6a 0a
 8048302: e8 05 00 00 00
 8048307: c9
 8048308: c3
 8048309: 90 90 90

0804830c <soma>:
 804830c: 55
 804830d: a1 74 93 04 08
 8048312: 89 e5
 8048314: 03 45 08
 8048317: a3 74 93 04 08
 804831c: c9
 804831d: c3
```

O *linker* resolve todas as referências a símbolos, trocando-as pelos seus endereços.

As instruções são representadas pelo seu código binário e não pelas suas mnemónicas.

Além do código correspondente à funcionalidade do programa de alto nível, o *linker* insere código necessário para lidar com o Sistema Operativo.

Sumário

Tema	Hennessy [COD]	Stallings [COA]	Bryant[CS:APP]
Arquitetura vs. Organização		Sec. 1.1, 2.2	
Máquina de von Neumann	Ver artigos de von Neumann e Godfrey na página da disciplina		
Stored Program	Sec 1.8	Sec. 2.1, 3.1	
Organização de von Neumann		Sec. 2.1, 3.1	
Tipos de instruções	Sec 3.1, 3.2, 3.5	Sec 2.1, 3.2	
Conversão entre níveis	Secs. 3.9, A1 ... A5		Sec. 3.2