Tópicos sobre Arquitectura de Computadores

José A. Cardoso e Cunha
DI-FCT/UNL

1. Arquitectura de Von Neumann
2. Programação em linguagem ´´máquina´´
3. Sistema de entradas e saídas
4. Hierarquia das unidades de memória
5. Organização interna do processador

1. Dado um problema
2. Um algoritmo para o resolver…
3. Que linguagens?
   - Para especificar o algoritmo
4. Que máquinas?
   - Para a execução do algoritmo

- Linguagens:
  - Mais próximas do utilizador humano
  - Mais próximas do computador

Programas

- Conjunto de instruções escritas numa linguagem que alguma máquina é capaz de reconhecer
- Instruções:
  - Mais próximas do domínio de aplicação: mais complexas e específicas
  - Mais próximas da arquitectura do computador: mais simples e genéricas
- Múltiplos níveis de linguagens
- Noção de interface
- Noção de transparência (ou opacidade)

Hierarquia de máquinas virtuais

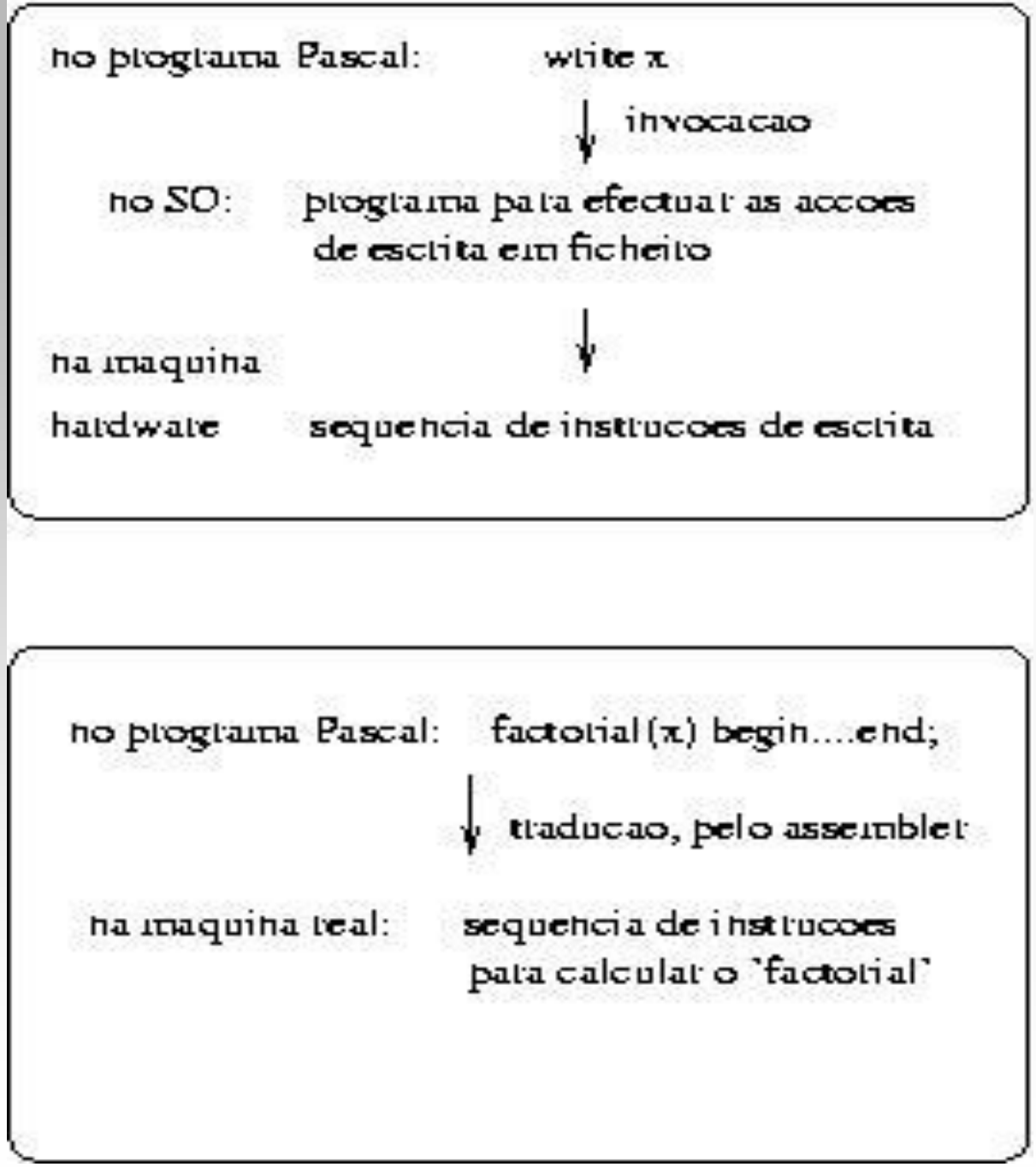
- Linguagens orientadas para a resol.problemas

- Sistema de operação
- Linguagem ´´assembly´

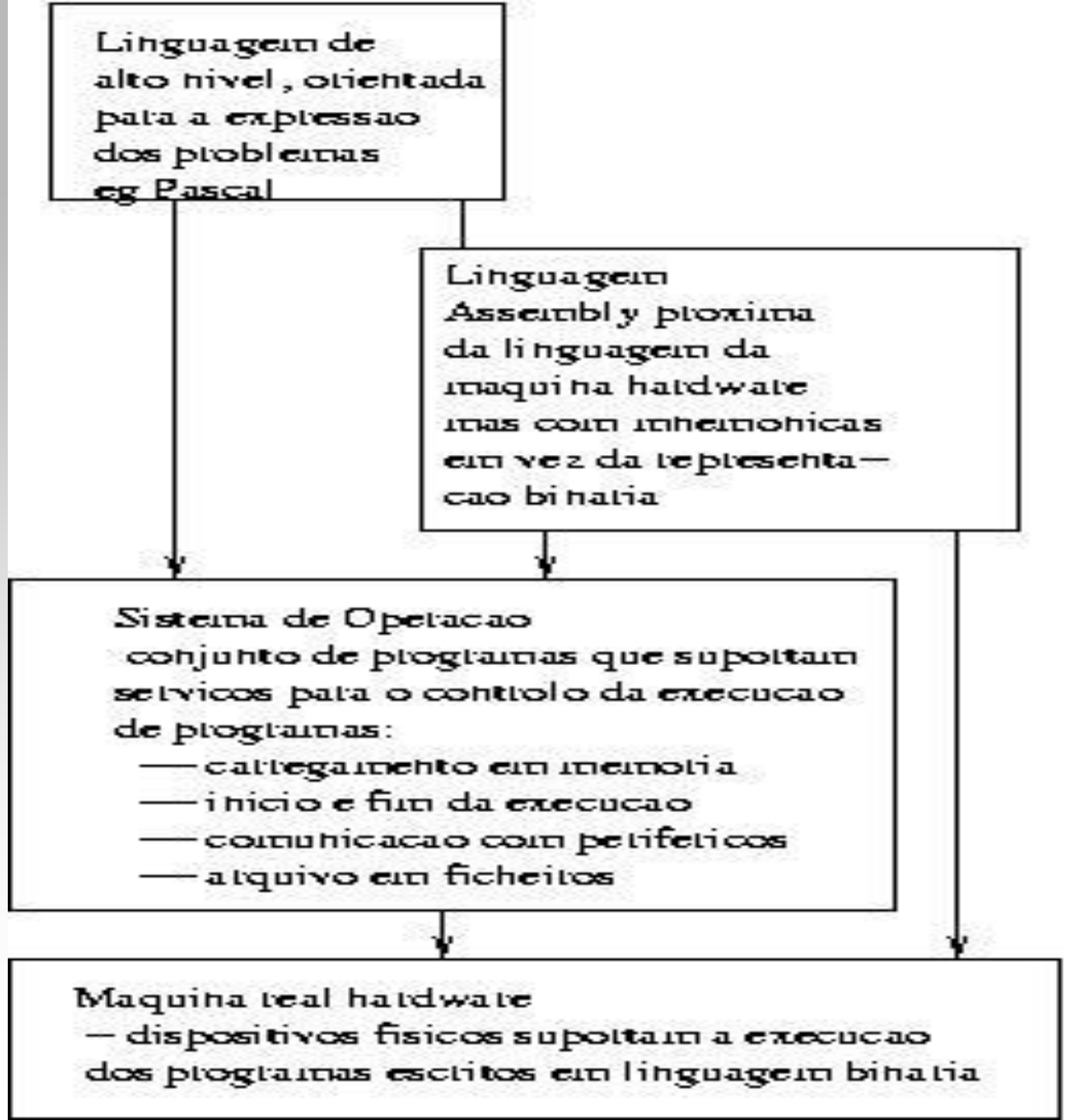
- Linguagem ´´máquina´´ (directamente suportada pelo ´´hardware´´do computador)

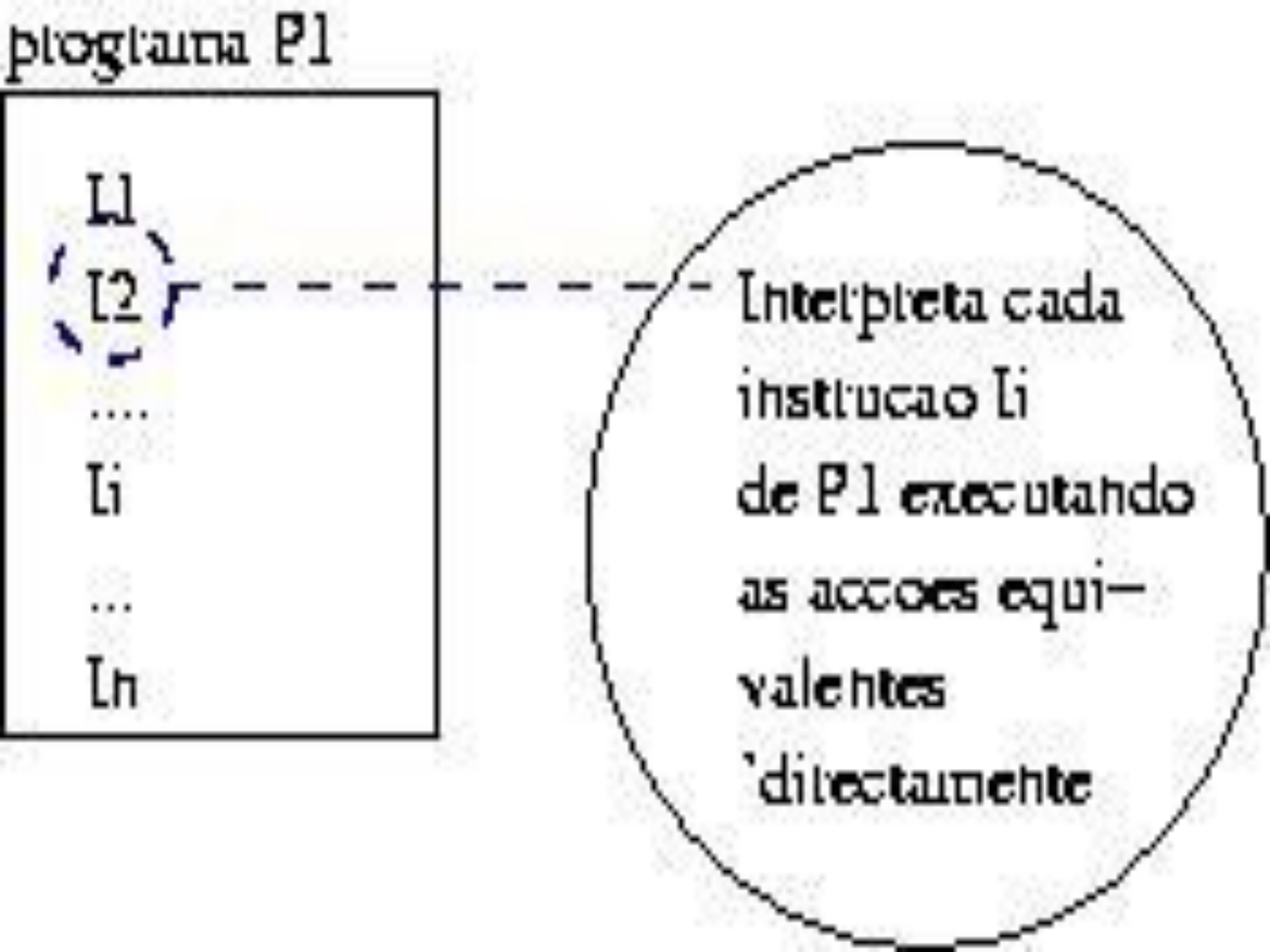
- Linguagem interna à micro-arquitectura
- Lógica digital


Programa
na linguagem
L
biblioteca de
suporte de L
Sistema de Operacao

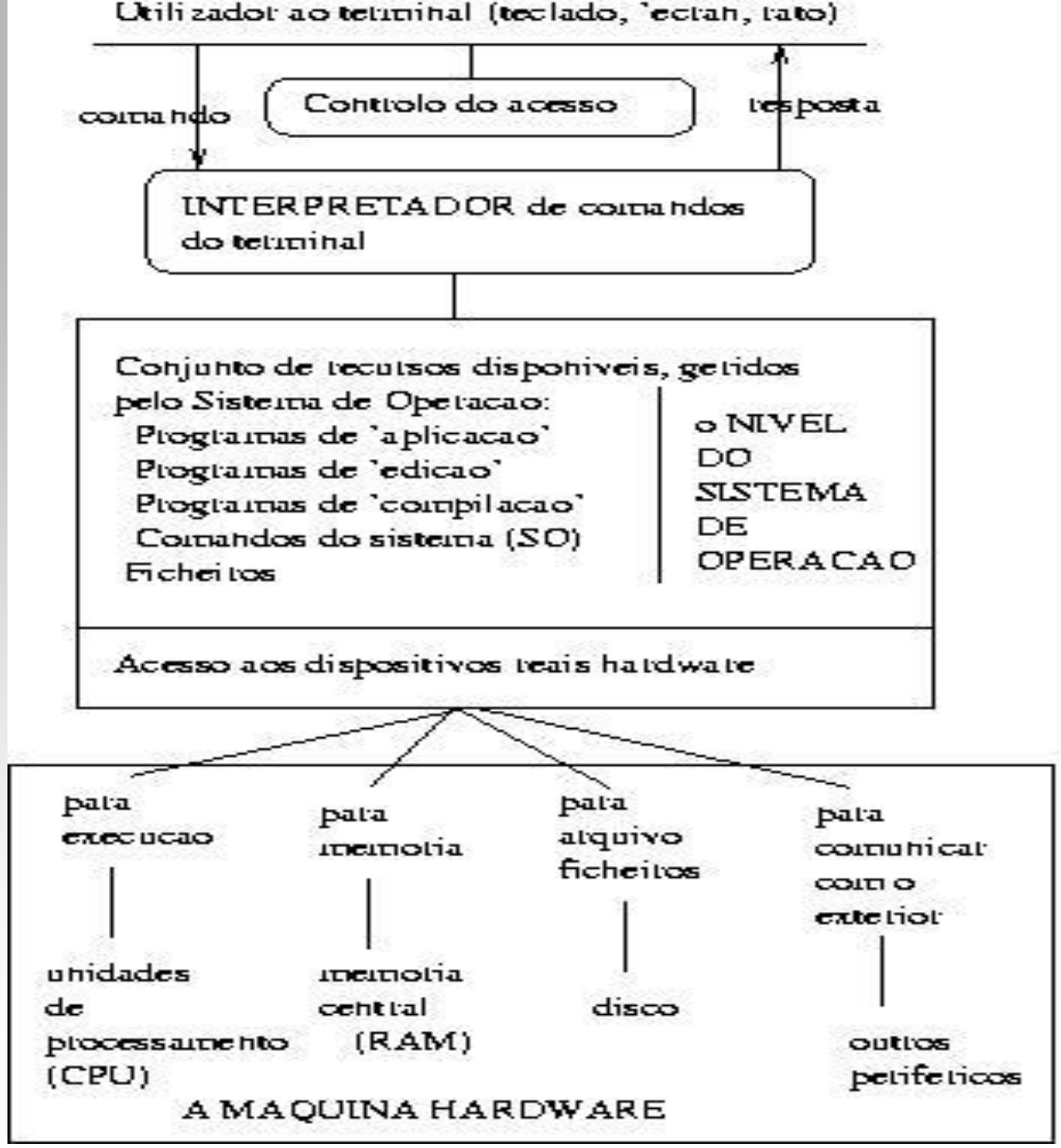

no programa Pascal: write x
invocacao
no SO: programa para efectuar as accoes de escrita em ficheiro
na maquina hardware
sequencia de instrucoes de escrita
no programa Pascal: factorial(x) begin....end;
traducao, pelo assembler
na maquina real: sequencia de instrucoes para calcular o `factorial`

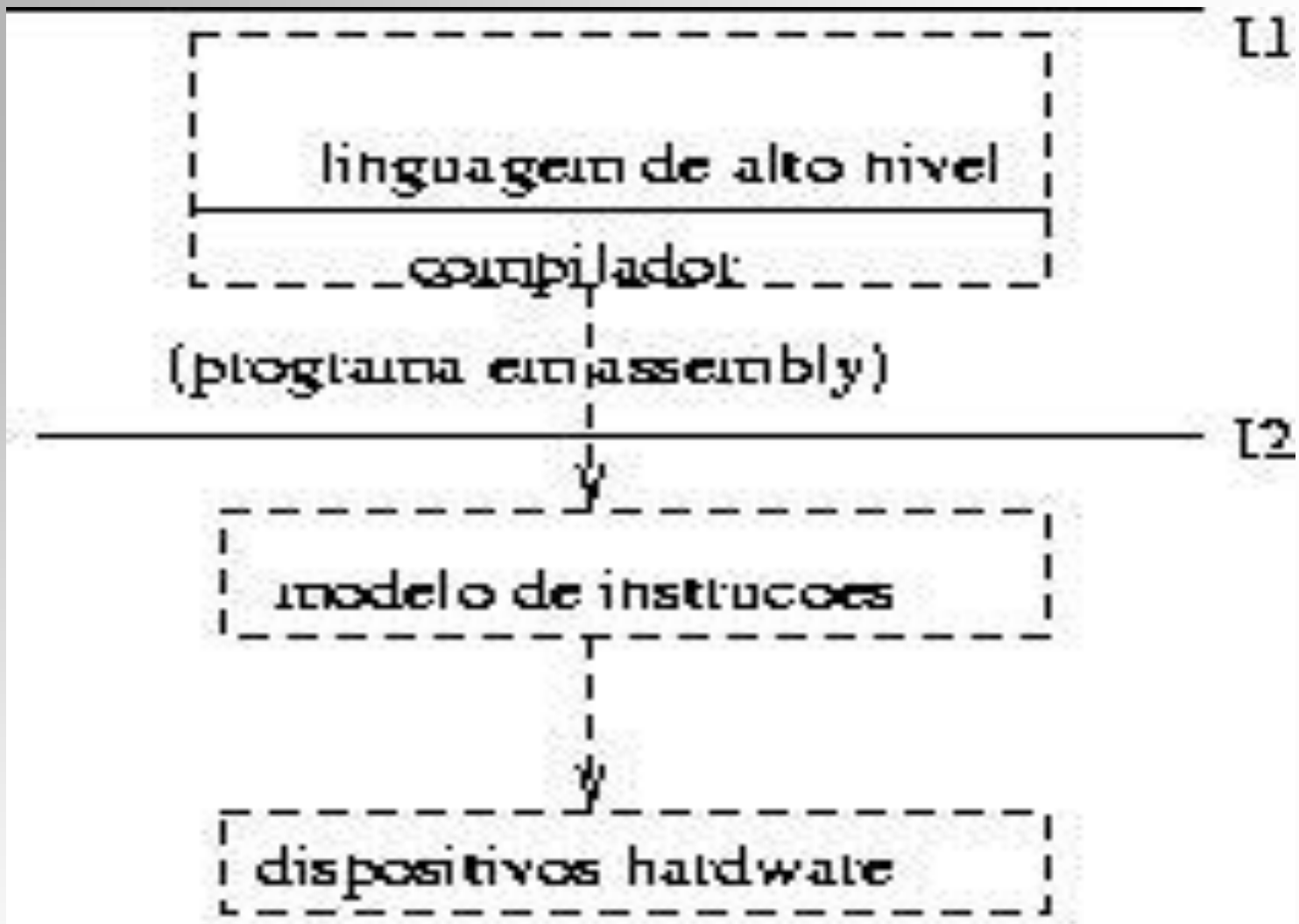
programa
P3
transforma
programa
P2


Linguagem de
alto nivel, otientada
pata a expiessao
dos pioblemas
eg Pascal
Linguagem
Assembly pioxima
da linguagem da
maquina haidwaie
mas com mnemonicas
em vez da iepiesenta—
cao binaiia
Sisteina de Opeiacao
conjunto de piogiamas que supoitam
seivicos pata o contiolo da execucao
de piogiamas:
— cailegamento em memoiia
— inicio e fim da execucao
— comunicacao com petifeiicos
— aiquivo em ficheiios
Maquina ieal haidwaie
— dispositivos fisicos supoitam a execucao
dos piogiamas escitos em linguagem binaiia


programa P1
L1
L2
....
Li
...
Ln
Interpreta cada
instrucao Li
de P1 executando
as accoes equi-
valentes
'directamente

Interpretador de comandos do SO

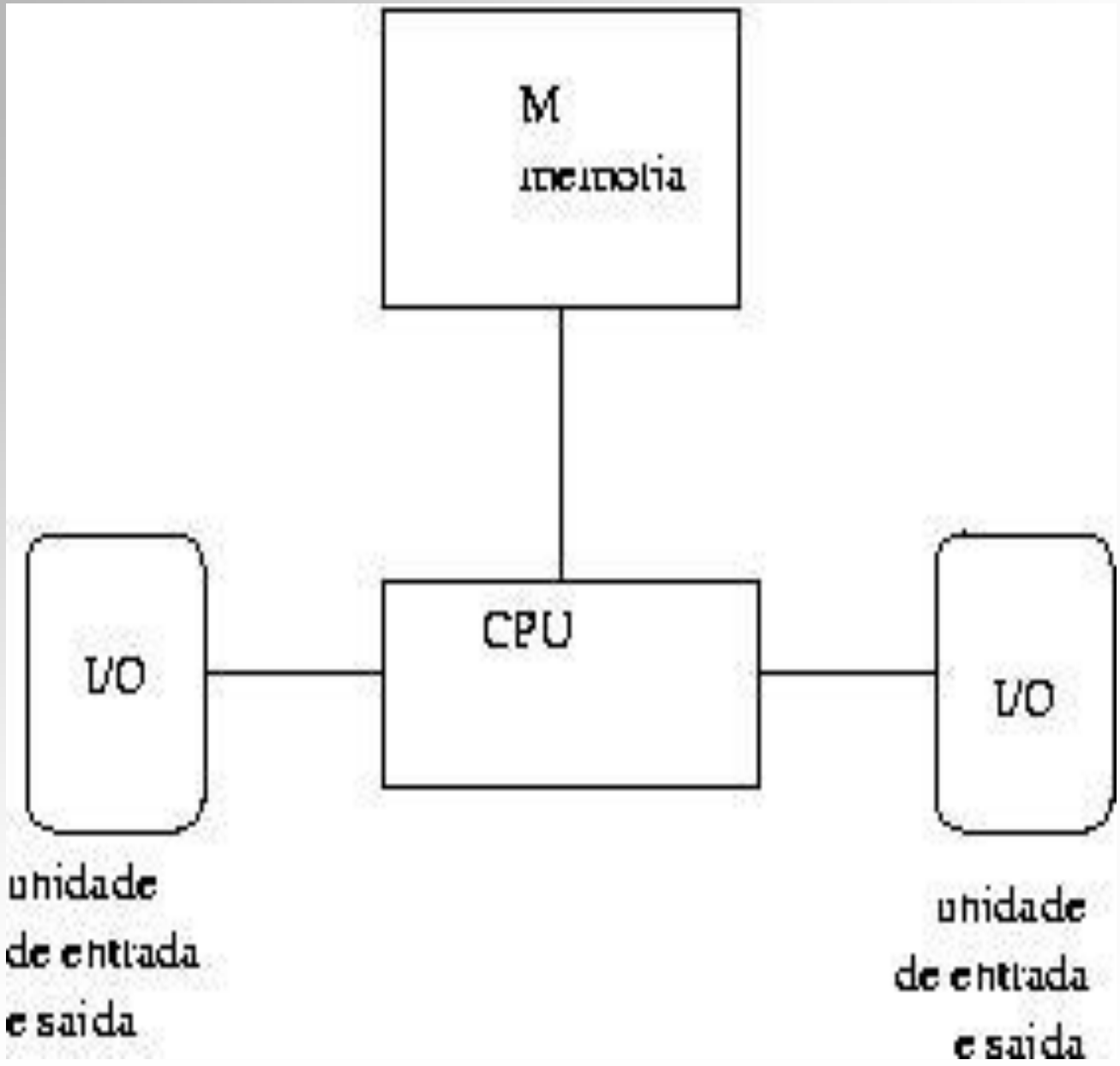

Utilizador ao terminal (teclado, ecran, rato)
comando
Controlo do acesso
resposta
INTERPRETADOR de comandos do terminal
Conjunto de recursos disponiveis, geridos pelo Sistema de Operacao:
Programas de 'aplicacao'
Programas de 'edicao'
Programas de 'compilacao'
Comandos do sistema (SO)
Ficheiros
o NIVEL DO SISTEMA DE OPERACAO
Acesso aos dispositivos reais hardware
para execucao
para memoria
para arquivo ficheiros
para comunicar com o exterior
unidades de processamento (CPU)
memoria central (RAM)
disco
outros perifericos
A MAQUINA HARDWARE

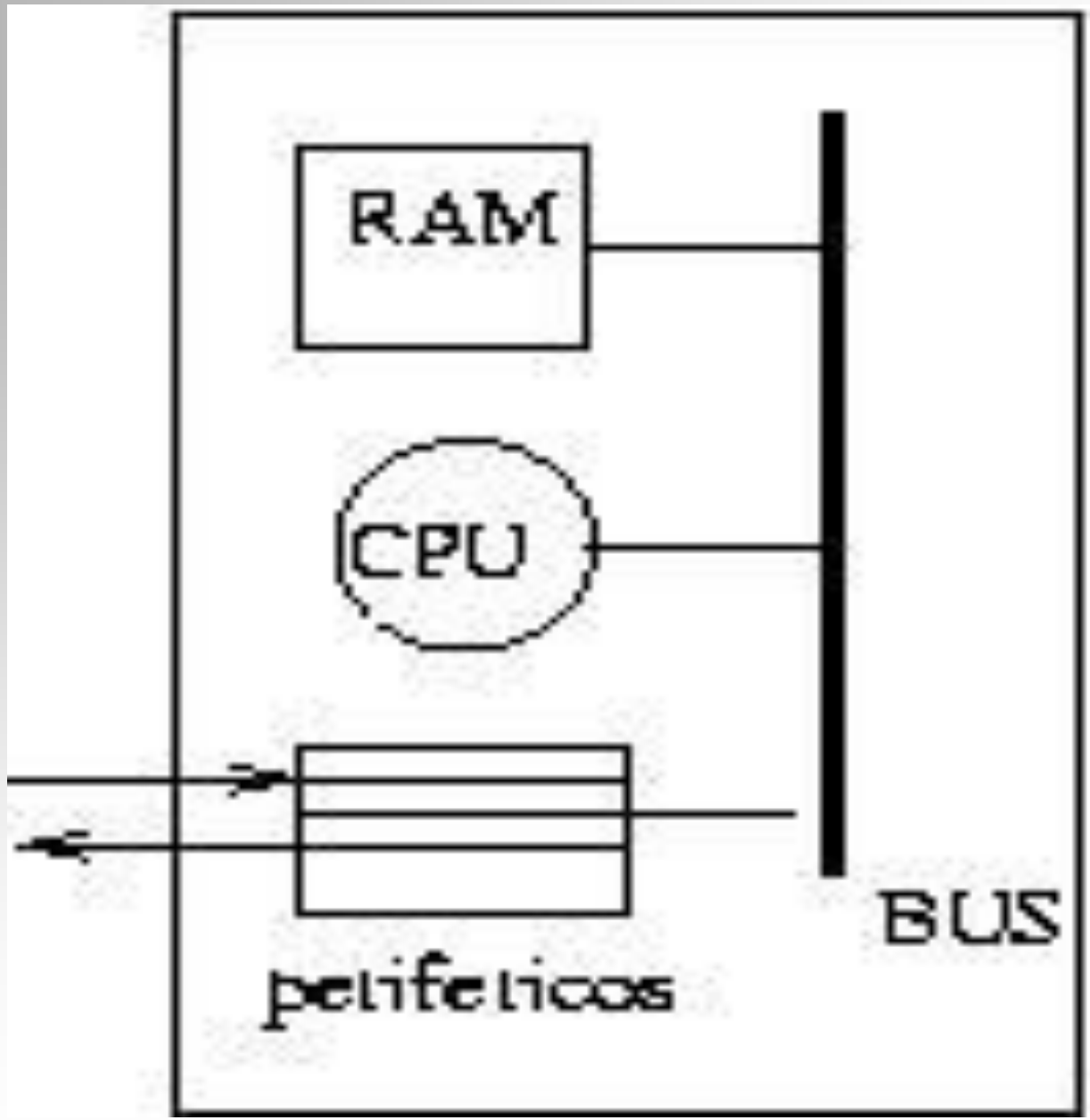


L1 – interface de linguagem de alto nível
L2 – interface de linguagem máquina

Arquitectura do computador

- Nível da ´´macro-arquitectura´´: (ISA)
  - Componentes: Processador, Memória, Unidades de Entrada/Saída, Interligações (*Bus*)
  - Linguagem máquina: representação binária de instruções e de dados (especificada sob a forma de mnemónicas, por conveniência: ´´assembly´´)
- Nível da ´´micro-arquitectura´´:
  - Componentes: portas lógicas, *flip-flops*, registadores, *buses*, *multiplexers*, *demultiplexers*, codificadores, descodificadores, contadores, unidades aritméticas e lógicas, memória
  - Realiza os componentes e as instruções da ´´macro-arquitectura´´


M
I/O
CPU
I/O
unidade
de entrada
e saida
unidade
de entrada
e saida


RAM
CPU
BUS
periféricos

Arquitectura de Von Neumann

Componentes (interligados através de buses):

- Processador central (CPU - *Central Processing Unit*)
- Memória central
- Unidades periféricas (armazenamento e comunicação)

Conjunto de instruções:

- Aritméticas e lógicas
- Transferências entre componentes
- Controlo da sequência de execução das instruções

Um modelo de execução muito simples:

- 1 instrução de cada vez: o Programa é uma sequência
- A Memória armazena as instruções e os dados
- 1 acesso a Memória de cada vez

Memória central

- Guarda instruções e dados, sob a mesma representação simbólica, em células que contêm agrupamentos de bits:

  bit, byte (8 bits), …

- É acedida como um vector de bytes:
  - Mem[0]…Mem[n-1] (capacidade = n bytes)
  - O índice i em Mem[i] chama-se o endereço da célula
  - O valor de Mem[i] chama-se o conteúdo da célula
  - O Acesso é directo: dá-se i para aceder a Mem[i]

  (*random access* – RAM: *Random Access Memory*)
  - O endereço é representado em binário, como um número inteiro sem sinal

Capacidade de memória

- O número máximo de bits do endereço determina a capacidade máxima de memória acessível
- Endereço Capacidade

  8 bits ---------- ?

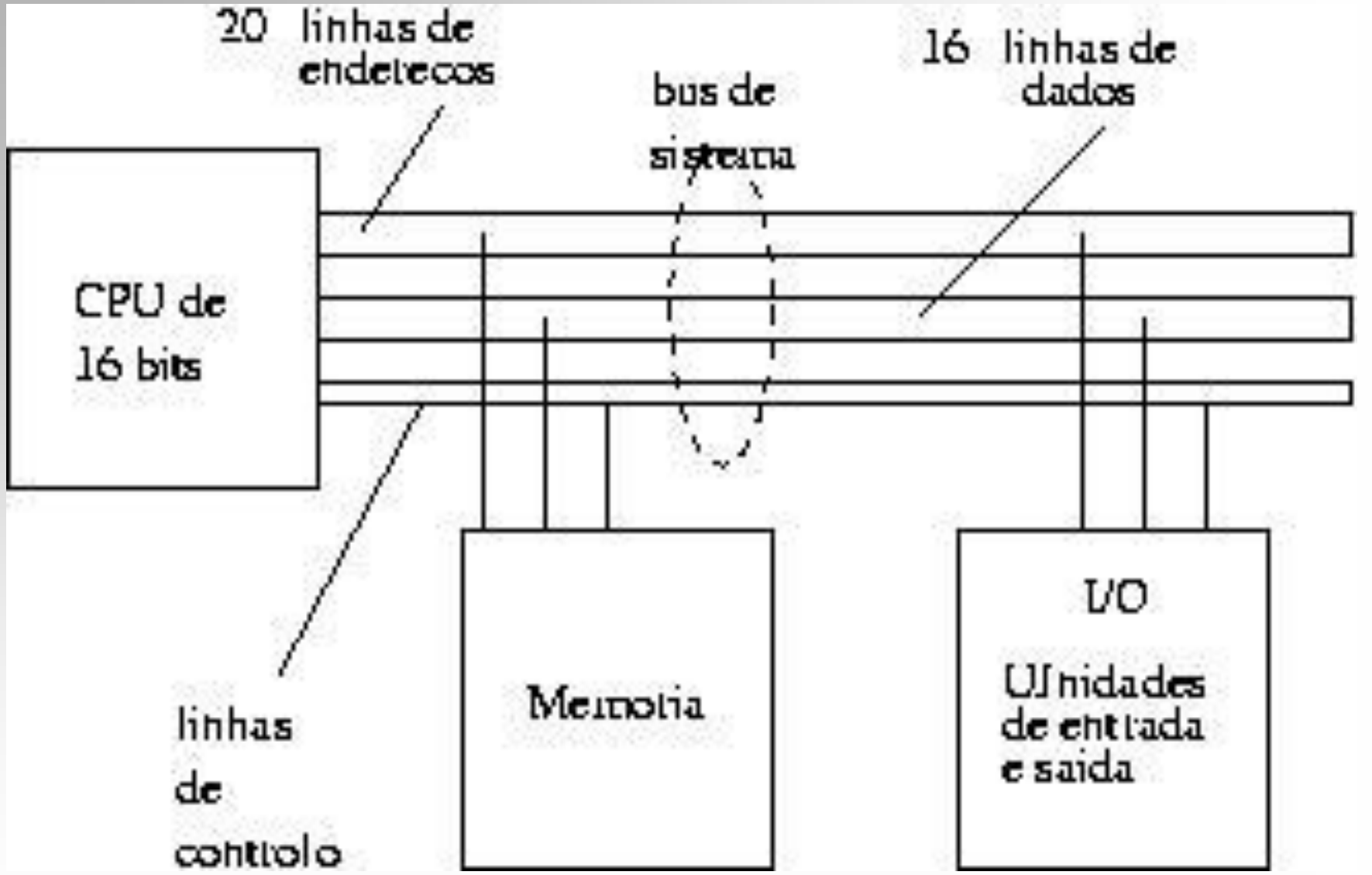
  16 bits

  20 bits

  24 bits

  30 bits

  32 bits
- Depende da memória e da arquitectura do computador

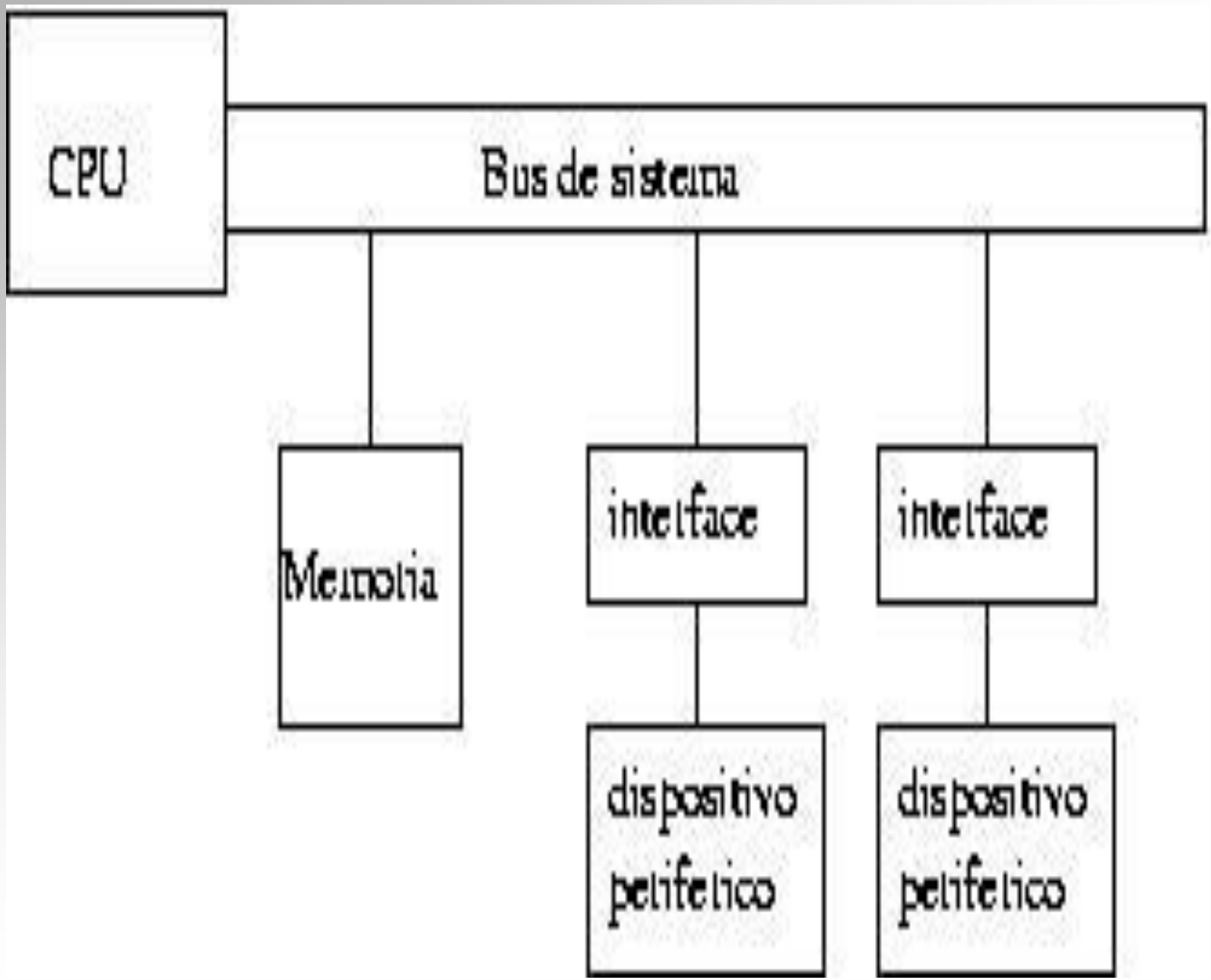

20 linhas de endereços
bus de sistema
16 linhas de dados
CPU de 16 bits
linhas de controlo
Memória
I/O
Unidades de entrada e saída

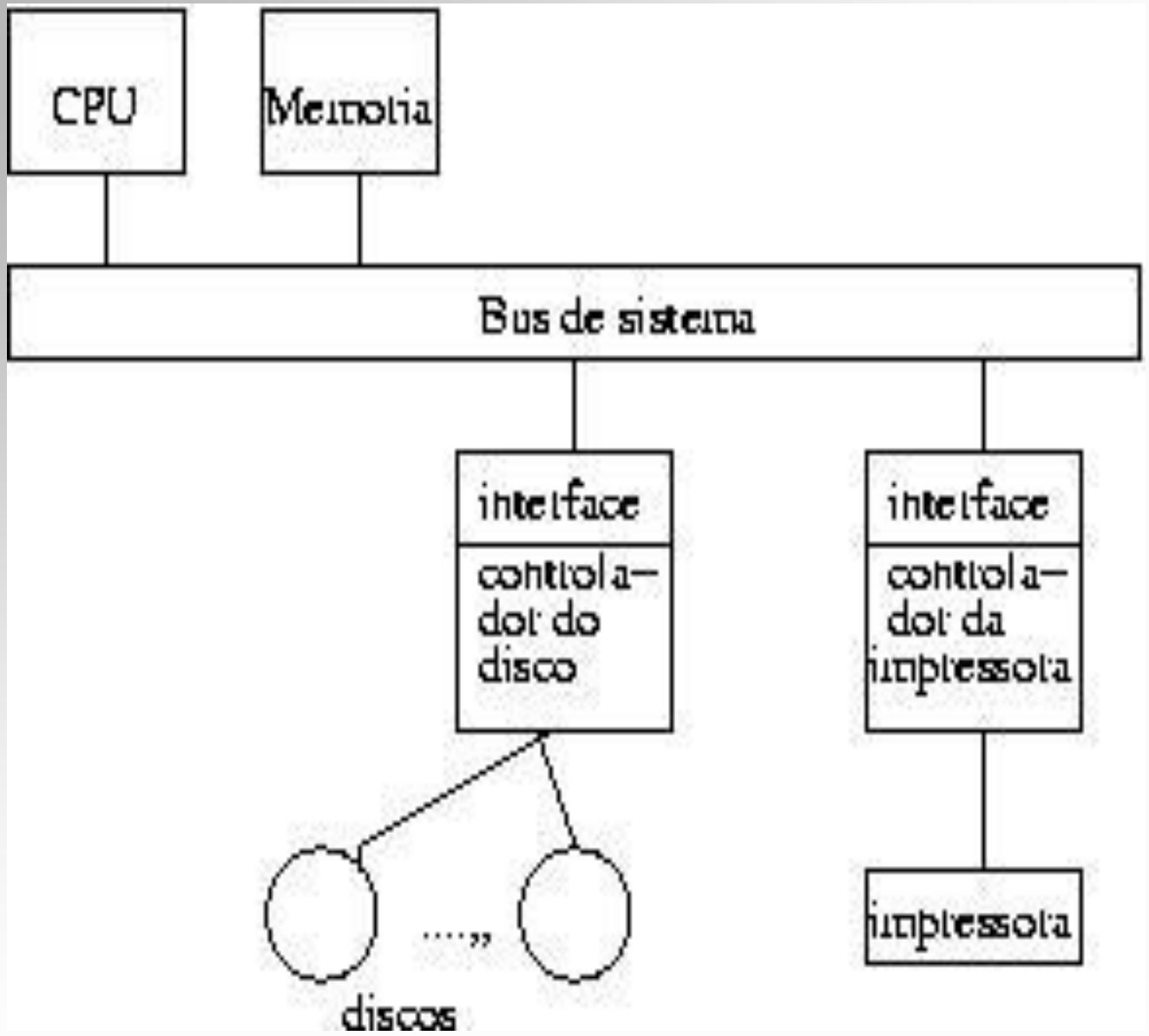
Processador - CPU

- Controlo global das operações do computador e responsável pela interpretação das instruções
- Contém:
  - Unidade de controlo: obtém, descodifica e interpreta as instruções (uma de cada vez)
  - Unidade aritmética e lógica: ALU – *Arithmetic and Logic Unit*
  - Conjunto de registadores: células de memória locais ao CPU, para valores intermédios

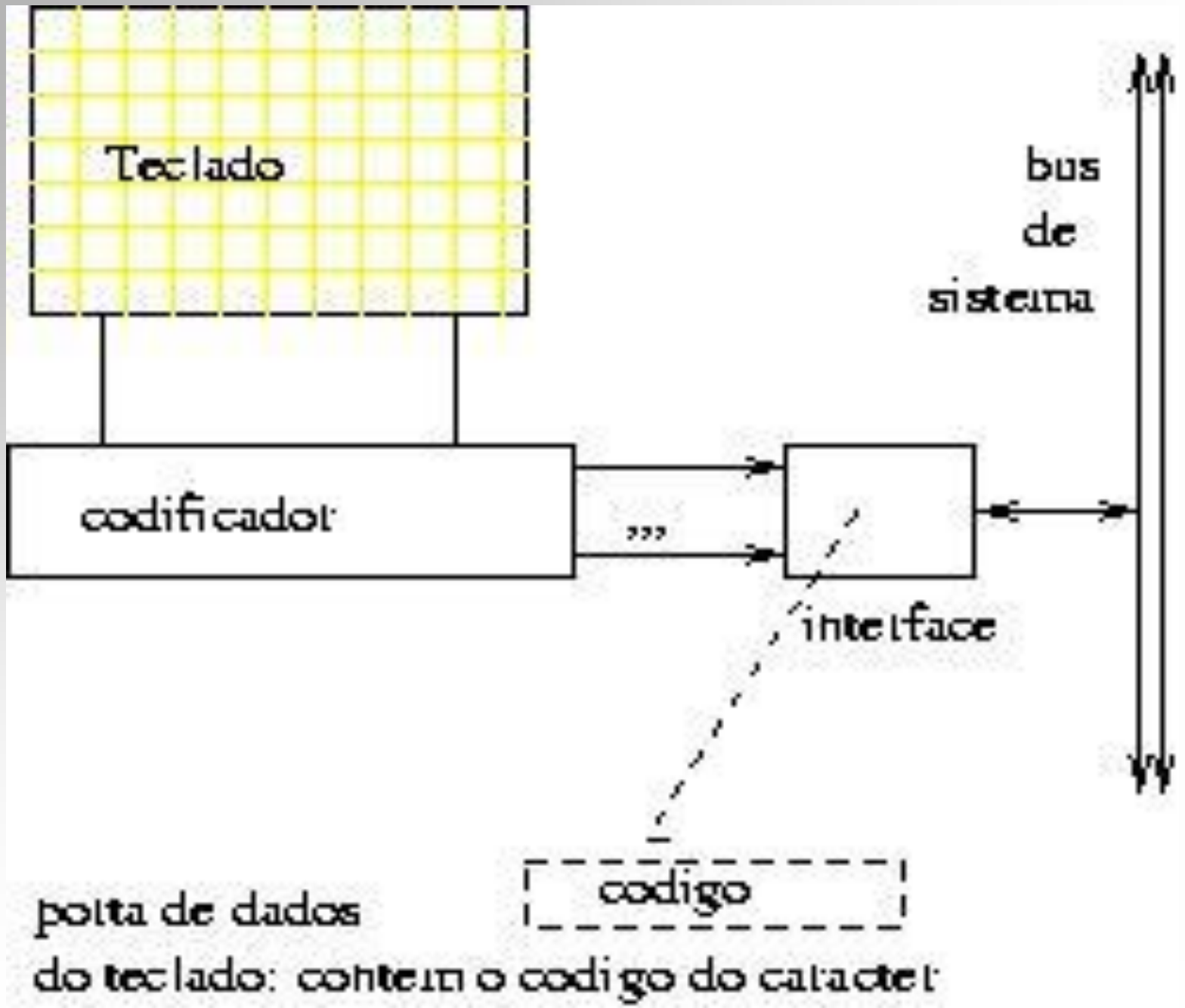
Unidades periféricas

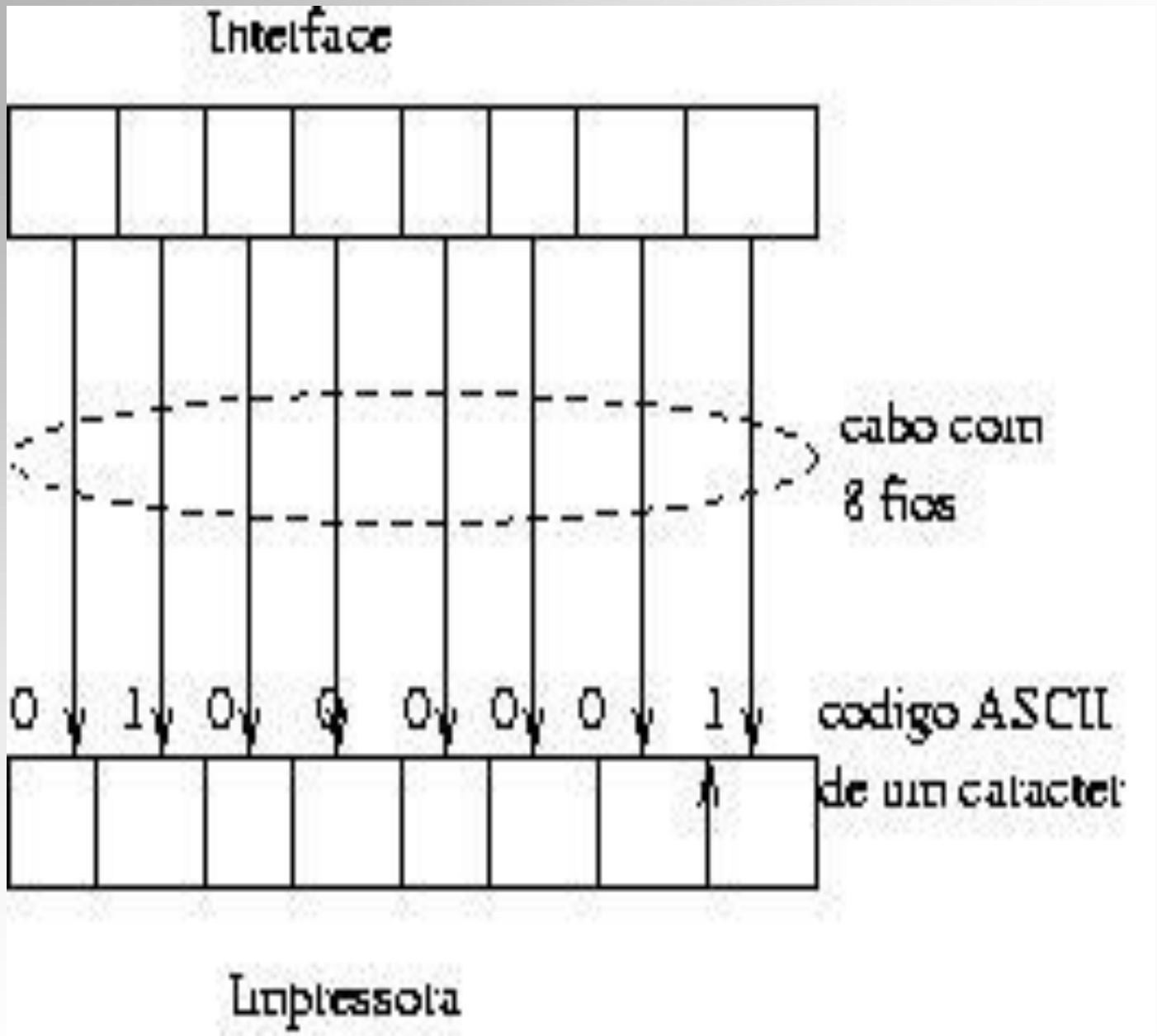
- Comunicação com o exterior:
  - teclado, écran, rato
  - impressora
  - portas de comunicação série e paralela (para ligação a rede de computadores e para uma diversidade de periféricos)
- Armazenamento de informação:
  - discos (Suporte de Memória Secundária)
  - outros dispositivos externos de suporte de armazenamento


CPU
Bus de sistema
Memoria
inteiface
inteiface
dispositivo
periferico
dispositivo
periferico


CPU
Memoria
Bus de sistema
interface
controla-
dor do
disco
interface
controla-
dor da
impressora
discos
impressora

Esquema genérico da interface


Teclado
bus
de
sistema
codificador
interface
código
porta de dados
do teclado: contem o codigo do caracter

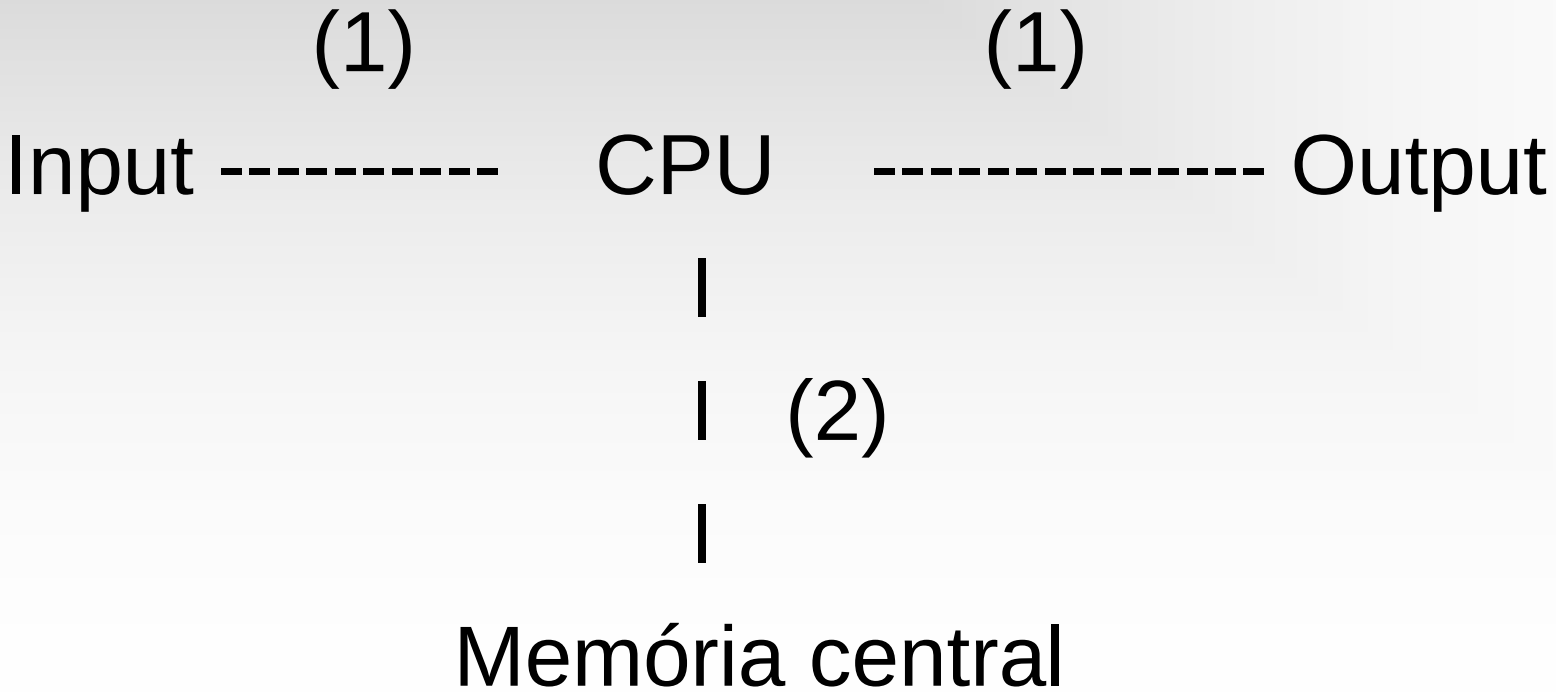

Interface
cabo com
8 fios
0 1 0 0 0 0 0 1
codigo ASCII
de um caractel
Impressora

Características da arquitectura

Operações dentro do CPU: muito rápidas

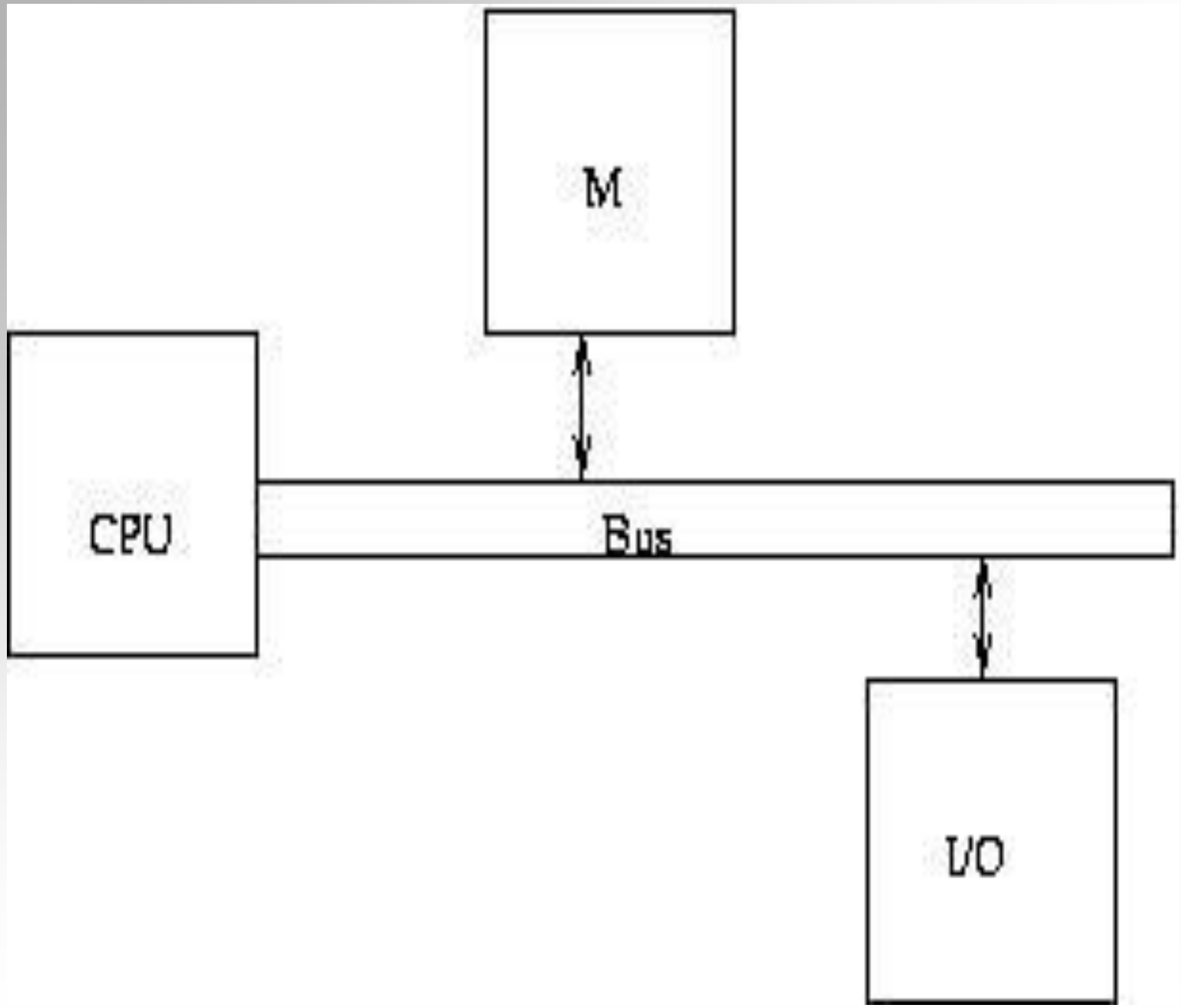
Interacções CPU – Memória: menos rápidas

Interacções com Periféricos: muito lentas



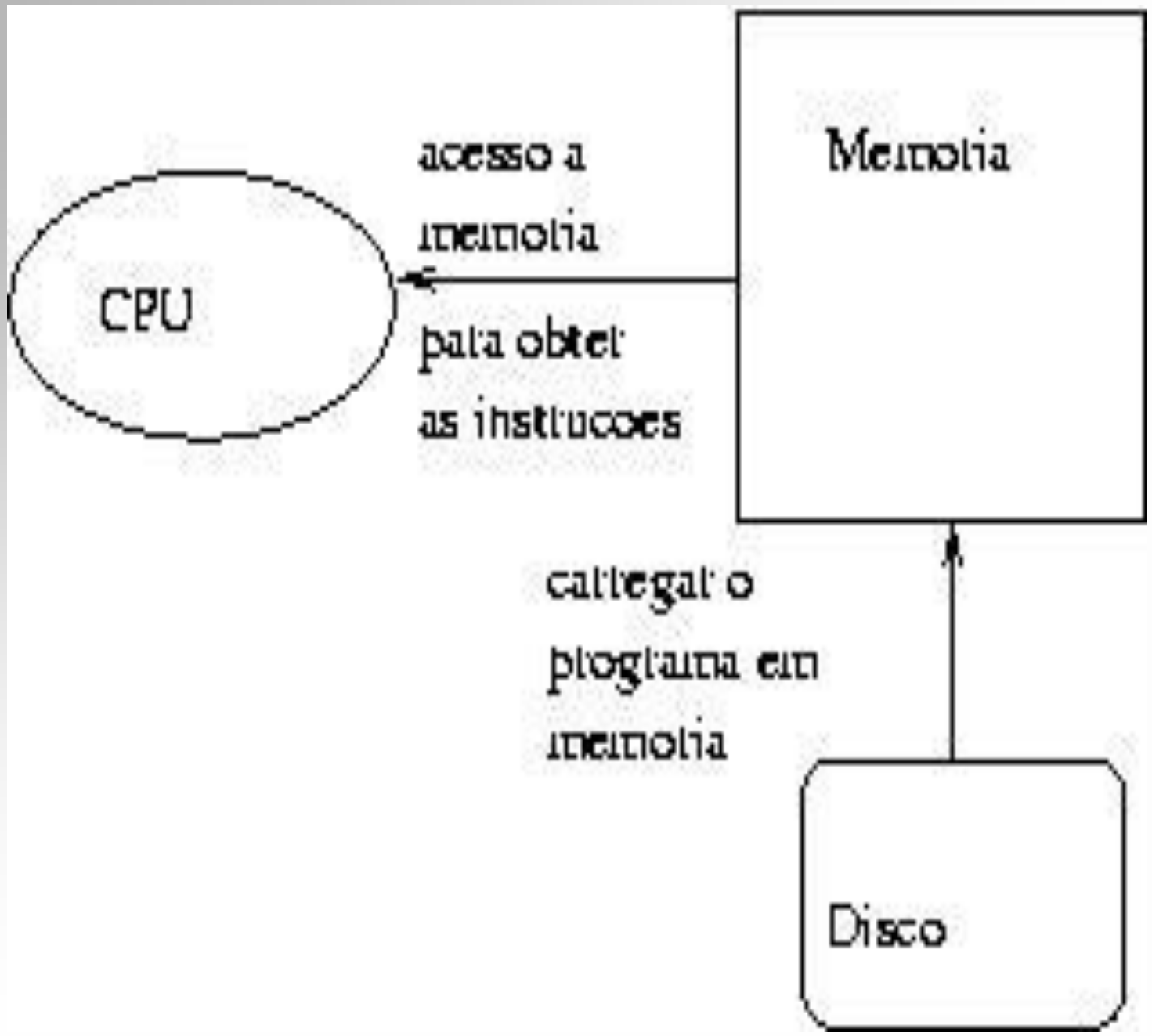
Topologias de buses

- Buses dedicados:
  - CPU – M
  - CPU – Periféricos
  - Periféricos – M
- Bus partilhado:
  - Um único bus interliga CPU, M e Periféricos


M
CPU
Bus
I/O

Modelo de execução de Von Neumann (década 1940)

- Máquina de uso geral: permite especificar programas que indicam as desejadas sequências de operações que deve executar
- Para ser genérica, o próprio algoritmo a executar, codificado em binário, como uma sequência de instruções, deve ser colocado na Memória central, para que o CPU possa iniciar a sua interpretação: *Program Stored Computer*
- ***A máquina executa qualquer programa que seja uma sequência válida de instruções***


CPU
acceso a
memoria
para obtener
las instrucciones
Memoria
cargar el
programa en
memoria
Disco

Modelo de execução de Von Neumann

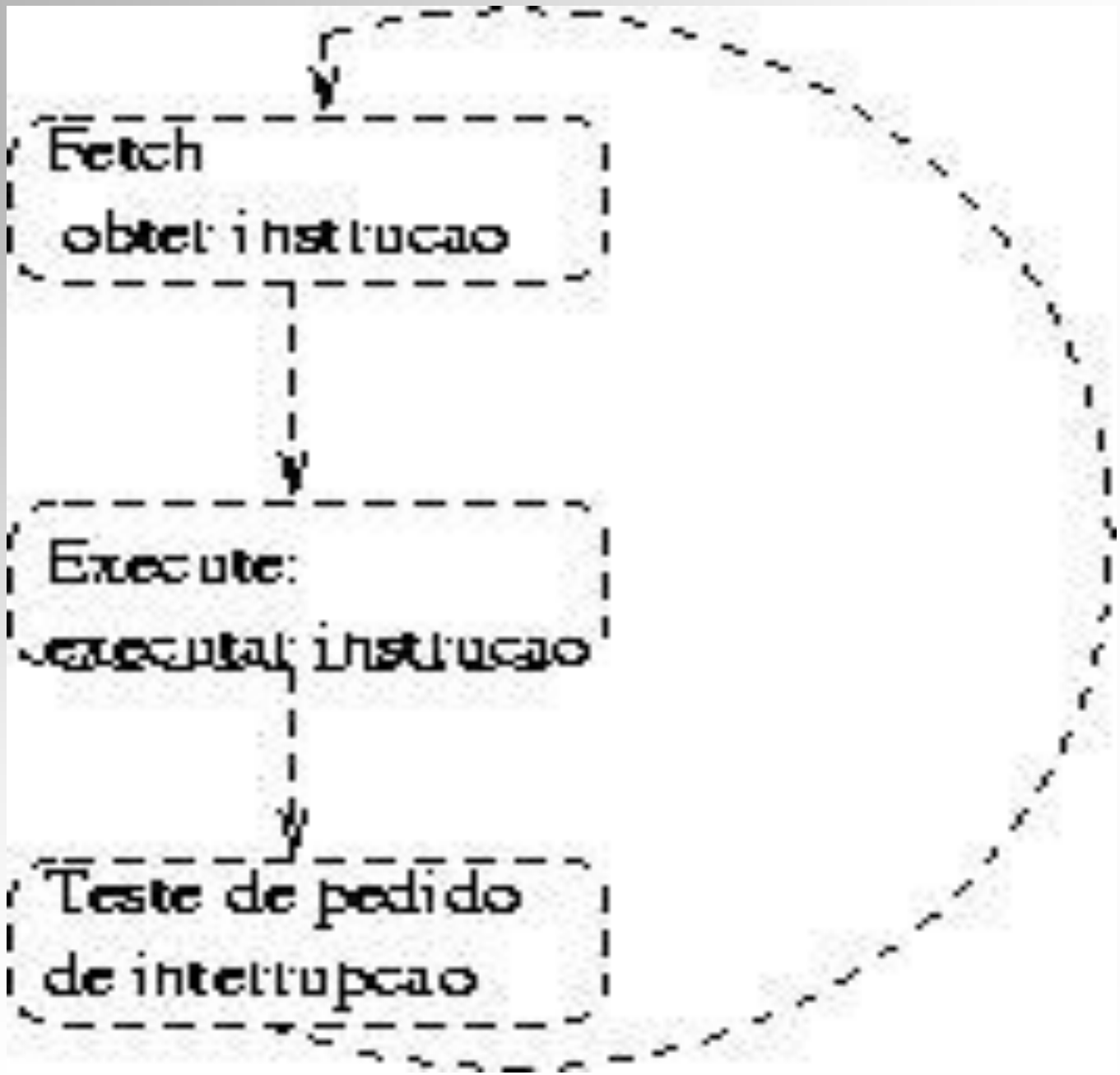
- Inicialmente,um programa é carregado em memória, como uma sequência de instruções, codificadas em binário, em posições consecutivas de memória (bytes)
- Após activado, o CPU inicia um ciclo eterno:

Enquanto ESTADO_ACTIVO fazer

(1) Obter instrução (Inst) de memória; – (*Fetch)*

(2) Executar instrução Inst; - (*Execute)*

- Ao executar a instrução *HALT*, o CPU termina o ciclo.


Fetch
obter instrucao
Execute:
executar instrucao
Teste de pedido
de interrupcao

A execução de Inst pode envolver:

- Operações aritméticas e lógicas (pela ALU)
- Transferências CPU – Memória
- Transferências CPU – Periféricos
- Controlo da sequência de execução de instruções

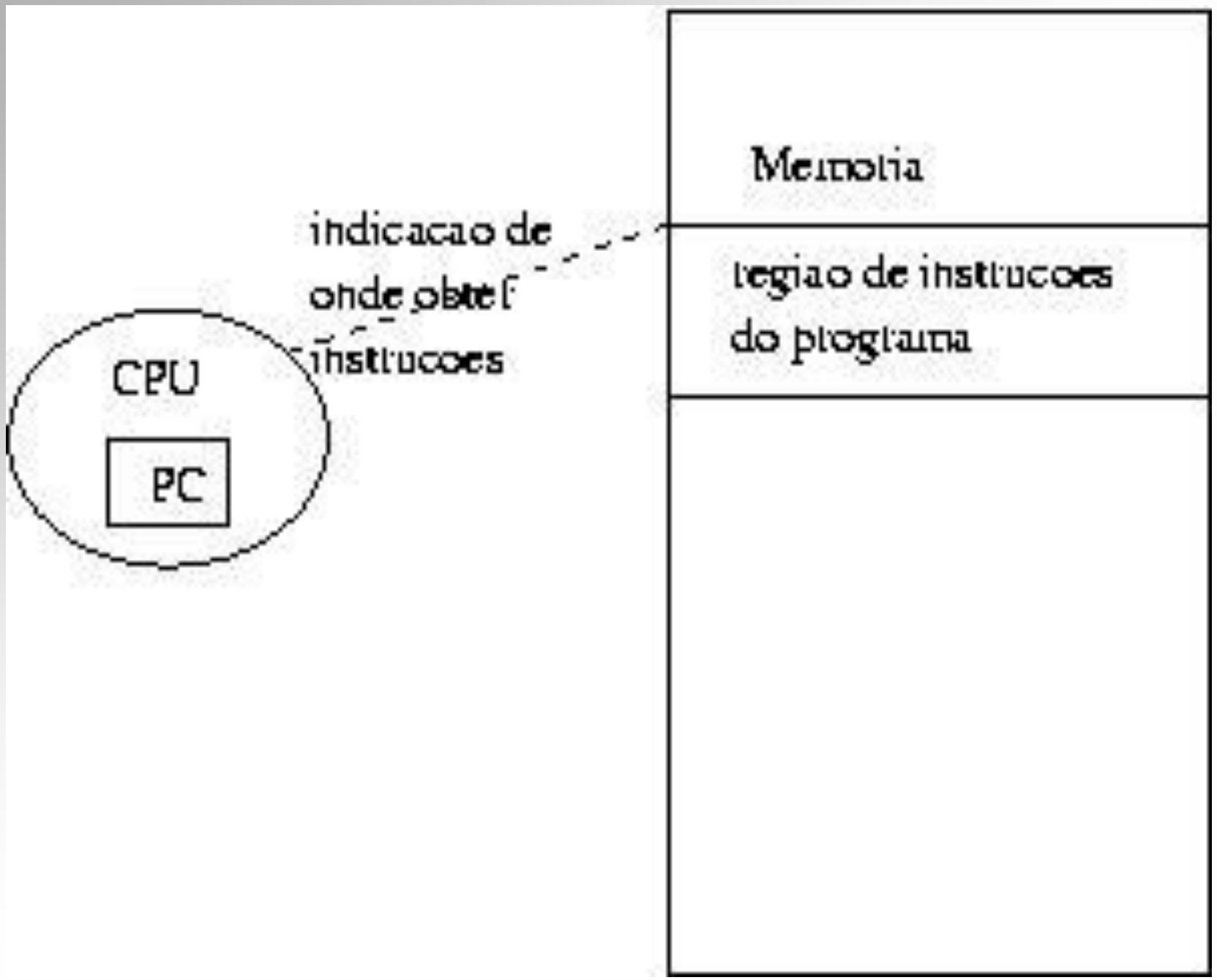
A ALU opera sobre números binários.

A Unidade de controlo do CPU:

- Obtém a próxima instrução de memória;
- Descodifica a instrução
- Emite os sinais de controlo, na micro-arquitectura, correspondentes ao encadeamento de acções necessárias para executar a instrução corrente e as transferências de informação necessárias.

Localizar a instrução em memória

- A que célula de memória deve o CPU ir buscar a instrução, no passo 1 (FETCH)?
- O programa foi previamente colocado em Memória, numa zona de células consecutivas, cujo endereço inicial é então conhecido.
- O CPU consulta um Registador especial – Program Counter (PC), que deve ter sido inicializado com o endereço de memória onde está a primeira instrução do programa:


Memoria
indicacao de
onde obter
instrucoes
regiao de instrucoes
do programa
CPU
PC

Obter uma instrução - FETCH

- Faz uma cópia da representação binária da instrução, a partir da memória, transferindo-a, pelo bus de sistema, para um registador especial do CPU: *Instruction Register (IR)*
- São acções automáticas da Unidade de controlo, no passo 1 do ciclo de execução
- A operação FETCH equivale a IR := Mem[PC]
- A instrução manter-se-á em IR durante toda a sua execução

Enquanto ESTADO_ACTIVO fazer

(1) Obter instrução (I) de memória; – (*Fetch)*

(2) Executar instrução I; - (*Execute)*

- Em pseudo-código:

Enquanto ESTADO_ACTIVO fazer

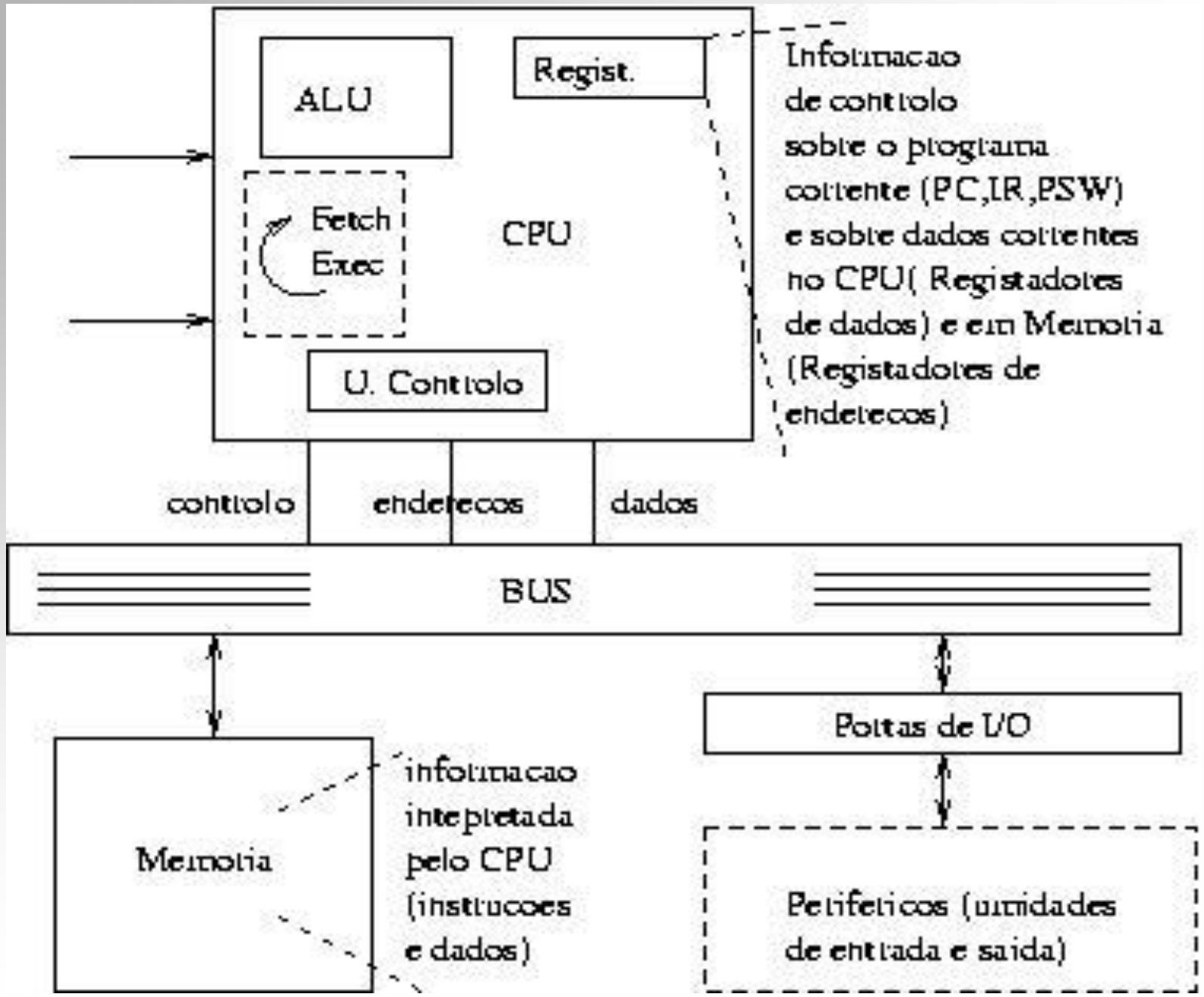
(1) IR := Mem[PC]; – (*Fetch)*

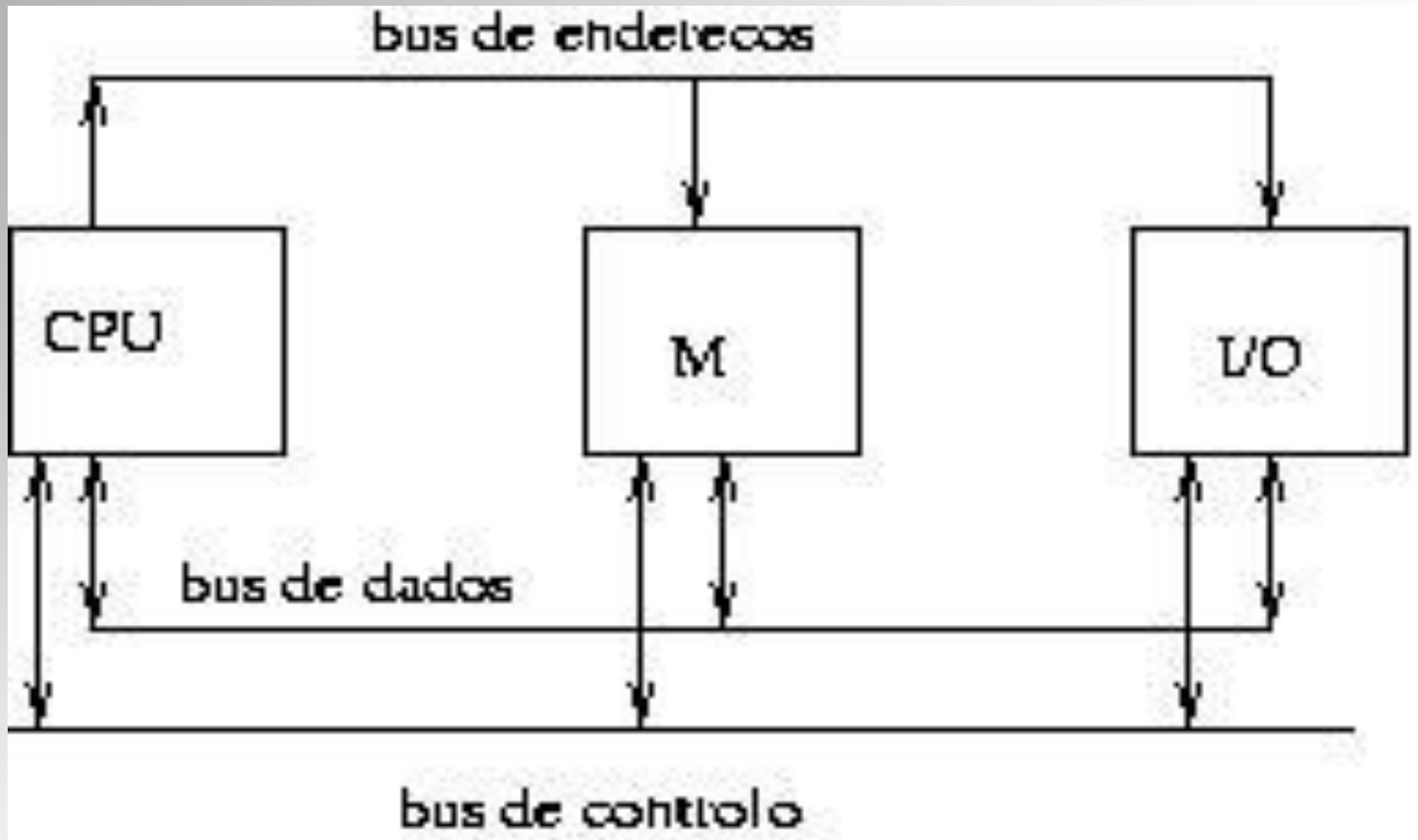
() PC := PC + 1; -- (preparar para próxima instrução)

(2) Executar instrução corrente I; - (*Execute)*

- Assumindo que:
  - A próxima instrução executar está na próxima célula de memória…
  - A representação de cada instrução ocupa 1 byte em memória…

Arquitectura de Von Neumann


ALU
Regist.
Fetch
Exec
CPU
U. Controlo
Informacao
de controlo
sobre o programa
corrente (PC,IR,PSW)
e sobre dados correntes
no CPU(Registadores
de dados) e em Memoria
(Registadores de
endereços)
controlo
endereços
dados
BUS
Portas de I/O
Memoria
informacao
interpretada
pelo CPU
(instrucoes
e dados)
Perifericos (unidades
de entrada e saida)



bus endereços – para identificar o dispositivo e a celula

bus dados – para transferir

bus controlo – para enviar sinais de controlo

- Bus de sistema: conjunto de linhas paralelas, cada transmite impulsos codificando um bit
- Bus de endereços:
  - conjunto de linhas que codificam o endereço
  - para identificar a célula de memória ou a unidade periférica
- Bus de dados:
  - codificam os dados a transferir
- Bus de controlo:
  - para coordenar as transferências e as interacções entre unidades, enviando comandos ou aguardando sinais de controlo

Interacções CPU - Memória

- O CPU ´´vê´´ a Memória como um vector de células M[0]…M[n]
- Cada célula contém uma unidade básica de informação: um grupo de bits
  - Em geral o endereço de memória refere-se a um byte: unidade mínima que se transfere entre CPU e memória
  - O número máximo de bytes, que se pode transferir de cada vez, depende do número de linhas do Bus de Dados:
    - Se Bus de dados tem 32 linhas: pode transferir 4 bytes

O índice de cada célula é codificado em binário como um endereço, que se coloca no Bus

O número de linhas de endereços do Bus determina a capacidade máxima de Memória Central que se pode aceder:

Espaço de Endereços Reais

Micro-operações de acesso à M

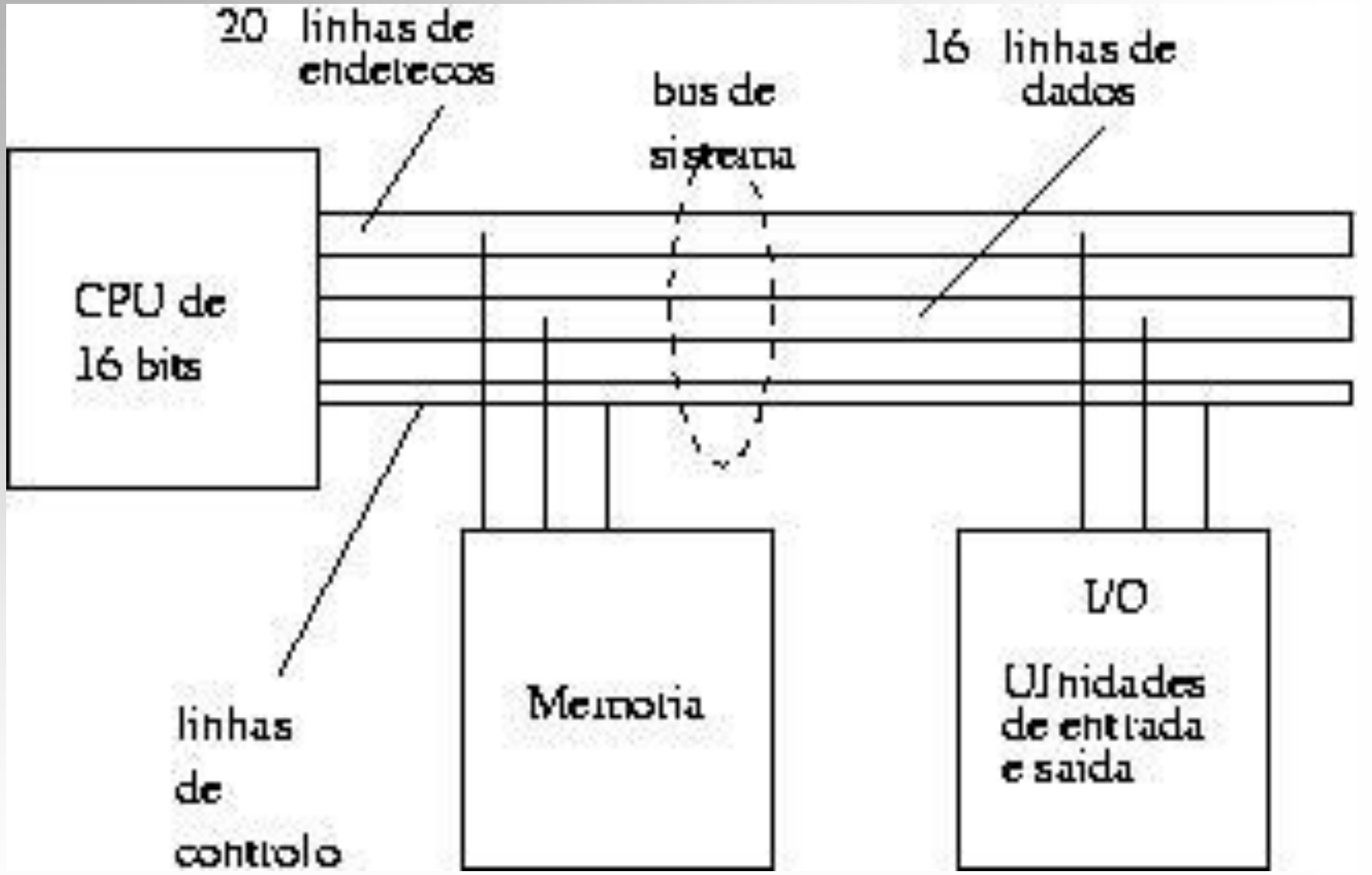
- O protocolo de interacção CPU – M assenta em duas operações elementares:

Em pseudo-código:

  - v := Ler(i) -- devolve em v, o conteúdo de Mem[i]
  - Escrever(v,i) – escreve o valor v em Mem[i]

O valor v fica à saída ou à entrada da unidade de memória, ligada ao Bus de sistema

- Estas micro-operações não são instantâneas.
- São automaticamente activadas pelo CPU quando precisa de ler ou escrever na memória:
  - Não são instruções máquina acessíveis ao programa


20 linhas de endereços
bus de sistema
16 linhas de dados
CPU de 16 bits
linhas de controlo
Memória
I/O
Unidades de entrada e saída

v := Ler(i)

- Para a efectuar, o CPU lança a sequência de acções:
  - Coloca o valor binário de i no bus de endereços
  - Coloca o código da operação ´´Ler´´ no bus de controlo
  - Aguarda até que o valor binário de ´´v´´ esteja presente no bus de dados, à saída da memória
  - Pode copiar o valor, do bus de dados, para um registador interno do CPU

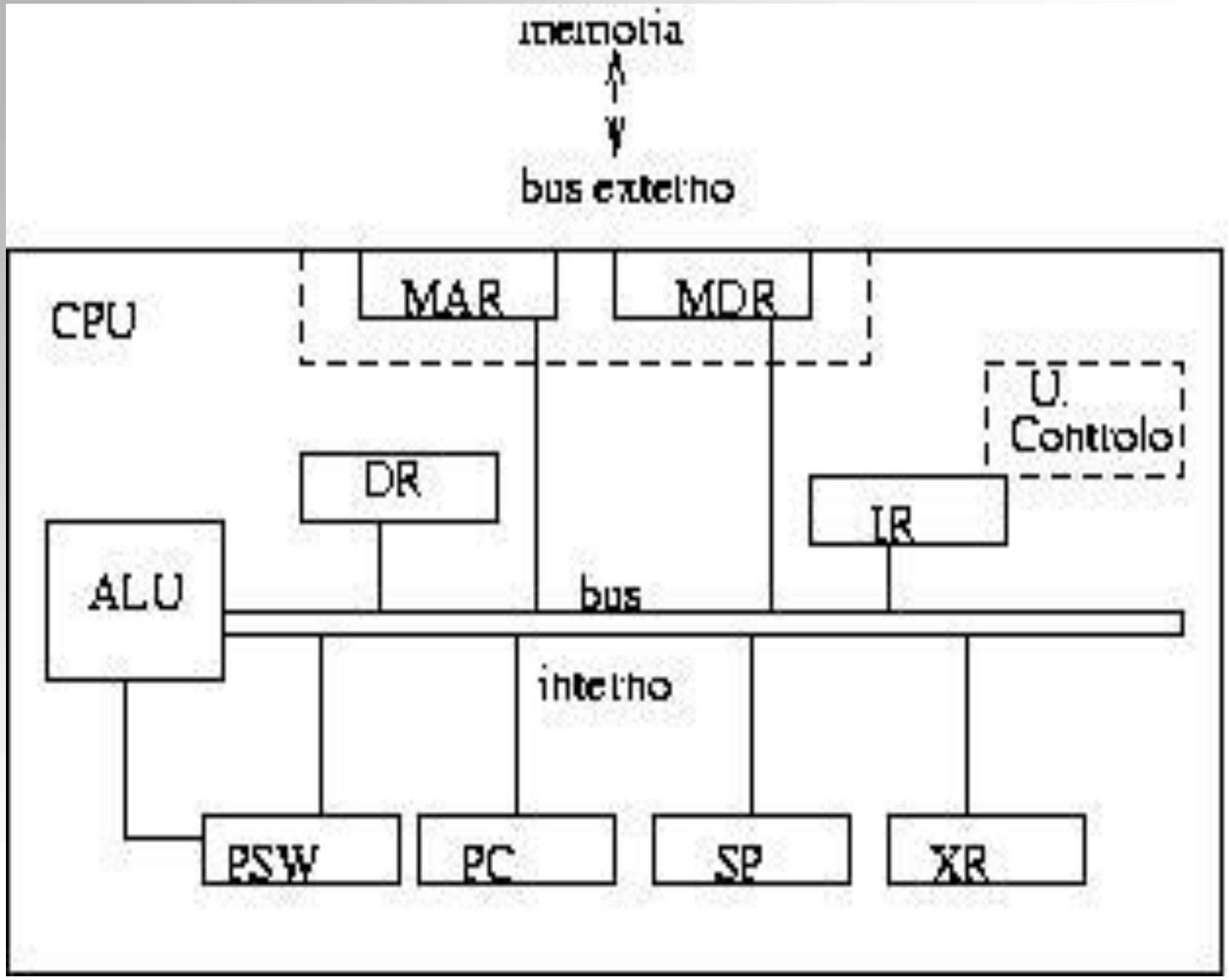
Escrever(v,i)

- Para a efectuar, o CPU lança a sequência de acções:
  - Coloca o valor binário de i no bus de endereços
  - Coloca o valor binário de ´´v´´ no bus de dados
  - Coloca o código da operação ´´Escrever´´ no bus de controlo
- Ao fim de um certo tempo, a unidade de memória modifica Mem[i] com o valor ´´v´´

- São automaticamente activadas pelo CPU quando precisa de ler ou escrever na memória:
  - Quando o CPU está no passo FETCH, para obter a próxima instrução a executar
  - Quando, durante a execução de uma instrução, o CPU vai buscar ou colocar dados a memória

- A memória opera sequencialmente:
  - Aceita operações de ler/escrever, uma de cada vez, com tempos de acesso bem definidos e constantes: Memória de tempo de acesso constante, seja qual for o endereço da célula
    - Em qualquer momento o CPU pode aceder a qualquer posição de memória: Random Access Memory

Estrutura interna do CPU


memoria
bus esterno
CPU
MAR
MDR
U. Controllo
DR
IR
ALU
bus
interno
PSW
PC
SP
XR

ALU: unidade aritmética e lógica

PSW (Processor Status Word) ou FR (Flags Register): informação do estado do CPU

IR (Instruction Register): código binário da instrução corrente em execução

PC (Program Counter) ou IP (Instruction Pointer): endereço da célula de Memória contendo a próxima instrução a executar

Unidade de controlo:

- Baseada no conteúdo de IR, descodifica a instrução e emite sinais de controlo para a sua execução;
- Controla também a execução do ciclo básico do CPU

Registos de Interface com o Bus (Bus Interface Unit): entre outros, contém:

- MAR (Memory Address Register)
- MDR (Memory Data/Buffer Register)
- Bus Control Register

Registos de trabalho do CPU:

- Um conjunto de Data Registers
  - para valores temporários
- Um conjunto de Address Registers
  - para mais eficiente acesso à memória

- Nota sobre o Bus interno do CPU
- Conceito de palavra interna ao CPU, relacionado com a ALU

Representação da informação

- Como? Onde?
- Bits e Bytes
- Bases de numeração binária (2), hexadecimal (16), decimal (10), octal (8)
- Valores inteiros:
  - Valores positivos em base 2:
    - $101_{(2)}$ -------- $5_{(10)}$
  - Valores positivos e negativos:
    - Que correspondência entre as sequências de bits e os valores? Duas representações:
    - A) Sinal mais Valor absoluto
    - B) Complemento a 2

	Sinal	- Valor absoluto
-3	1	11
-2	1	10
-1	1	01
-0	1	00
+0	0	00
+1	0	01
+2	0	10
+3	0	11

Domínio –3 a +3

Complemento a 2

-4	1	00
-3	1	01
-2	1	10
-1	1	11
0	0	00
+1	0	01
+2	0	10
+3	0	11

Domínio –4 a +3

Número negativo $-x$ $(x>0)$

Com uma representação em n bits

Em ´´complemento a 2´´ é a representação em base 2, do número

$$2^n - x$$

Exemplo: $n = 3$

$2^n = 2^3 = 8$ -- 1000

$x = 2$ -- 010 (número 2)

$2^n - x = 6$ -- 110 (número -2)

Complemento a 2 1000

n 010

2 - x 110 (-2)

Complemento a 1 111

n 010 (2)

(2 - 1) - x 101

Complemento a 2 = (compl. a 1) +1

= (101)

1

110

Exemplo: n = 8 representação com 8 bits

Domínio – 2^{n-1} $2^{n-1} - 1$

10000000 01111111

-128(10) +127(10)

Números negativos: bit (n-1) a 1

Números positivos: bit (n-1) a 0

dito o bit de sinal

Números reais

Notação científica:

-4 ← expoente

$3.2 * 10$ 0.00032

mantissa

Vírgula flutuante (*Floating point*)

$m * 2^n$

m e n – representados por sequências de bits separadas

Normas internacionais definem o formato:

(IEEE)

Exemplo; representação com 16 bits

10 bits – mantissa (bits 15 a 6)

6 bits -- expoente (bits 5 a 0)

$$1.25 \text{ ---- } 5 * 2^{-2}$$

m = 5 ---- 0000000101

n = -2 ---- 111110

empacotados como:

0000 0001 01 11 1110

0 1 7 E em base 16

Caracteres

Codificados segundo códigos normalizados:

ASCII – American Standard Code for Information Interchange

7 bits

Cada carácter: representação binária de um número de 0 a 127

Exemplo: ´A´ -- $01000001_{(2)}$ -- $41_{(H)}$ -- $65_{(10)}$

´%´-- $00100101_{(2)}$ -- $25_{(H)}$ -- $37_{(10}$

+ caracteres de controlo

(estendido para 8, dá +128 caracteres)

Exemplo: um controlador de écran ou de uma impressora recebem sequências de 8 bits e mostram o carácter com o código ASCII correspondente:

$Y = 9778_{(10)} - \underline{00100110}\ \underline{00110010}_{(2)}$ --2632 (H)

Se enviar isto, aparece ´&´ seguido de ´2´

O programa deve enviar os códigos adequados, de ´9´, ´7´ , ´7´, ´8´

Variáveis de linguagens de ´´alto nível´´

Aos nomes das variáveis de programas numa linguagem de alto nível, correspondem:

- Uma representação binária
- Uma localização em memória

Uma variável é representada por um certo número de bytes, dependendo de:

- Tipo da variável
- Compilador
- Arquitectura do computador

Exemplo:

Var X, Y, Z: integer; -- 4 bytes cada

Var U,V: char; -- 1 byte cada

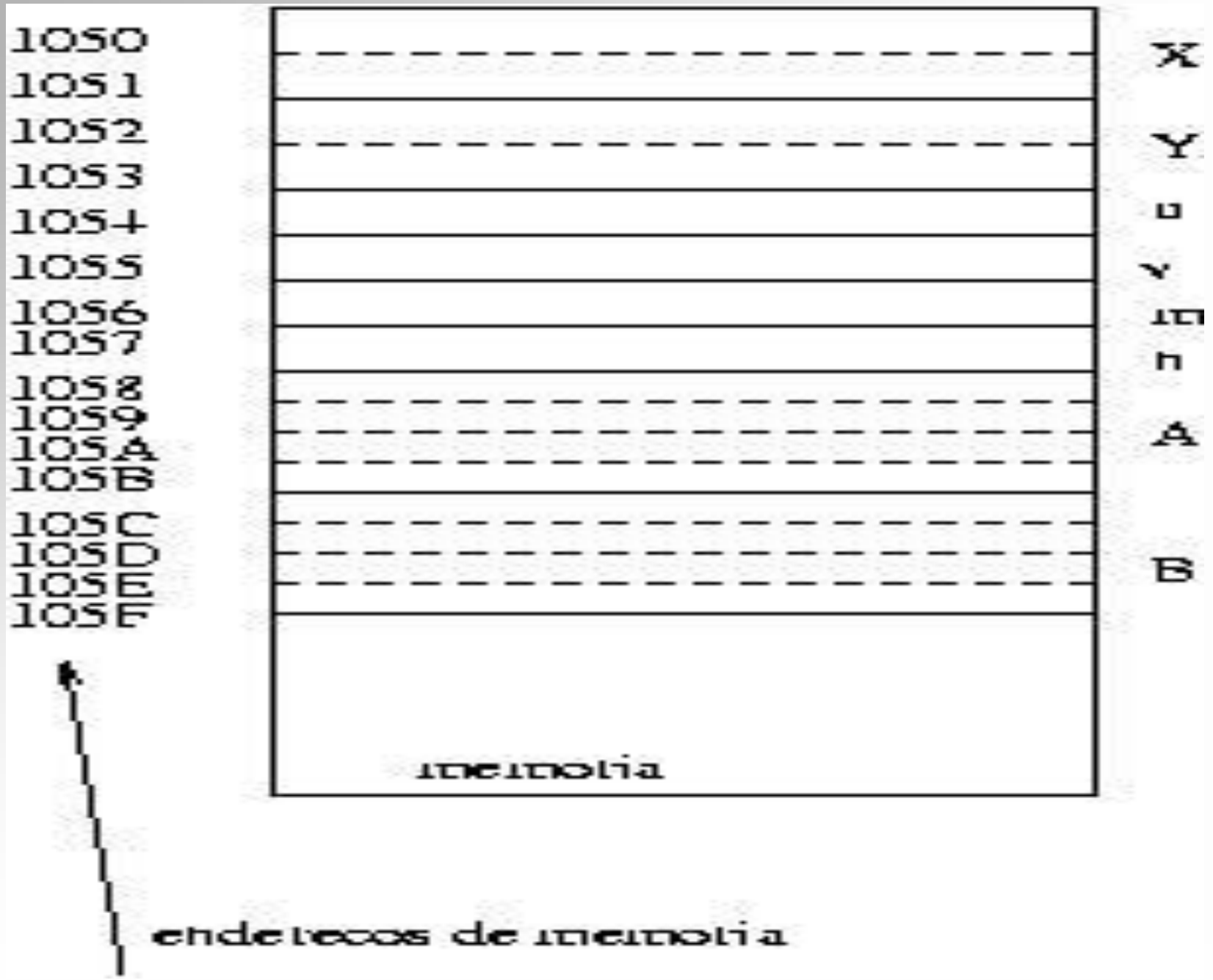
Var M,N: boolean; -- 1 byte cada

Var A,B,C: real; -- 4 bytes cada

Na memória, as variáveis aparecem representadas numa memória linear: um vector de bytes:

Variável	endereço
x	20050
y	20054
z	20058

etc.


105O
105S1
105S2
105S3
105S4
105S5
105S6
105S7
105S8
105S9
105SA
105SB
105SC
105SD
105SE
105SF
X
Y
endereços de memória

Instruções máquina

- Um grupo de bits codifica cada instrução e é reconhecido pelo CPU
- Sequências de instruções guardadas em memória são interpretadas pelo ciclo de execução do CPU

A memória central armazena:

- O Programa: sequência de instruções em células consecutivas de memória, codificadas em binário
- ´´Dados´´: valores contidos em células de memória, em binário, que são interpretadas pelo CPU como os operandos das instruções máquina, ou os resultados da sua execução

Coexistem na mesma memória física…

Então:

11101111

Será uma instrução ou um dado?

Arquitectura de um computador

Caracterizada por:

- Conjunto de instruções do processador
- Modelo de memória (alcance de um programa, alcance que o bus permite, capacidade de memória)
- Estrutura interna do processador (conjuntos de registadores, etc)
- Topologias de interligação dos componentes CPU, memória e periféricos (estruturas de buses)

- Um mesmo conjunto de instruções pode ter distintas realizações: noção de *família* de processadores, com custos e desempenhos variáveis

Conjunto de instruções do CPU

- Programa: sequência de instruções, em células consecutivas de memória
- Instrução: definição de uma operação elementar, capaz de ser executada pelo CPU

1. Que tipo de instruções se devem incluir na arquitectura?
2. Que informação deve incorporar-se no código de cada instrução?

As classes de instruções necessárias são poucas:

- Aritméticas e lógicas
- Transferências
- Controlo da sequência de execução

Os objectivos a considerar são muitos:

- Aproximar das linguagens de alto nível
- Reduzir o tamanho dos programas
- Oferecer instruções poderosas e eficientes
- Oferecer diversos modos de endereçar as variáveis em memória (ex. int, arrays, records)
- Reduzir o número de acessos a memória central

- Instruções simples,que facilitem realização eficiente
- Oferecer só as instruções de facto úteis
- Promover uma abordagem em que instruções mais complexas não são oferecidas, mas podem ser implementadas, por programas, de forma eficiente

Para as boas soluções, contribuem:

- A definição do conjunto de instruções
- E todos os outros elementos da arquitectura…

A solução boa é também fruto de compromissos:

- Custo
- Funcionalidade
- Desempenho
- Leis do mercado e circunstâncias tecnológicas…

Que instruções incluir: (lista não exaustiva)

- Multiplicação e divisão
- Aritmética de vírgula flutuante
- Aritmética de dupla precisão
- Chamada-retorno de funções/métodos
- Execução eficiente de ciclos e iterações
- Contar e temporizar acções
- Operar sobre cadeias de caracteres, sobre listas, sobre filas, sobre records, sobre matrizes
- Mover e recolocar programas em memória
- Suspensão e reactivação de programas
- ????

A que nível: na arquitectura do computador ou em níveis superiores da hierarquia?

Informação a incluir na instrução

Depende do tipo de instrução:

- O código da operação
- A especificação dos operandos ou sua localização
- A especificação dos resultados (localização)

Exemplo: formato de instrução com 4 campos:

código – end1 -- end2 – end3

Para operações com dois operandos (em end1 e end2), deixando o resultado em end3)

´´end´´ pode indicar uma célula de memória ou um registador do CPU

Quantos bits para o código?

Depende do número de instruções distintas

Exemplo: 8 bits → 256 instruções possíveis

Quantos bits para os endereços?

Depende de:

- onde estão os operandos,
- o modo como se endereça a memória
- a capacidade do espaço de endereços

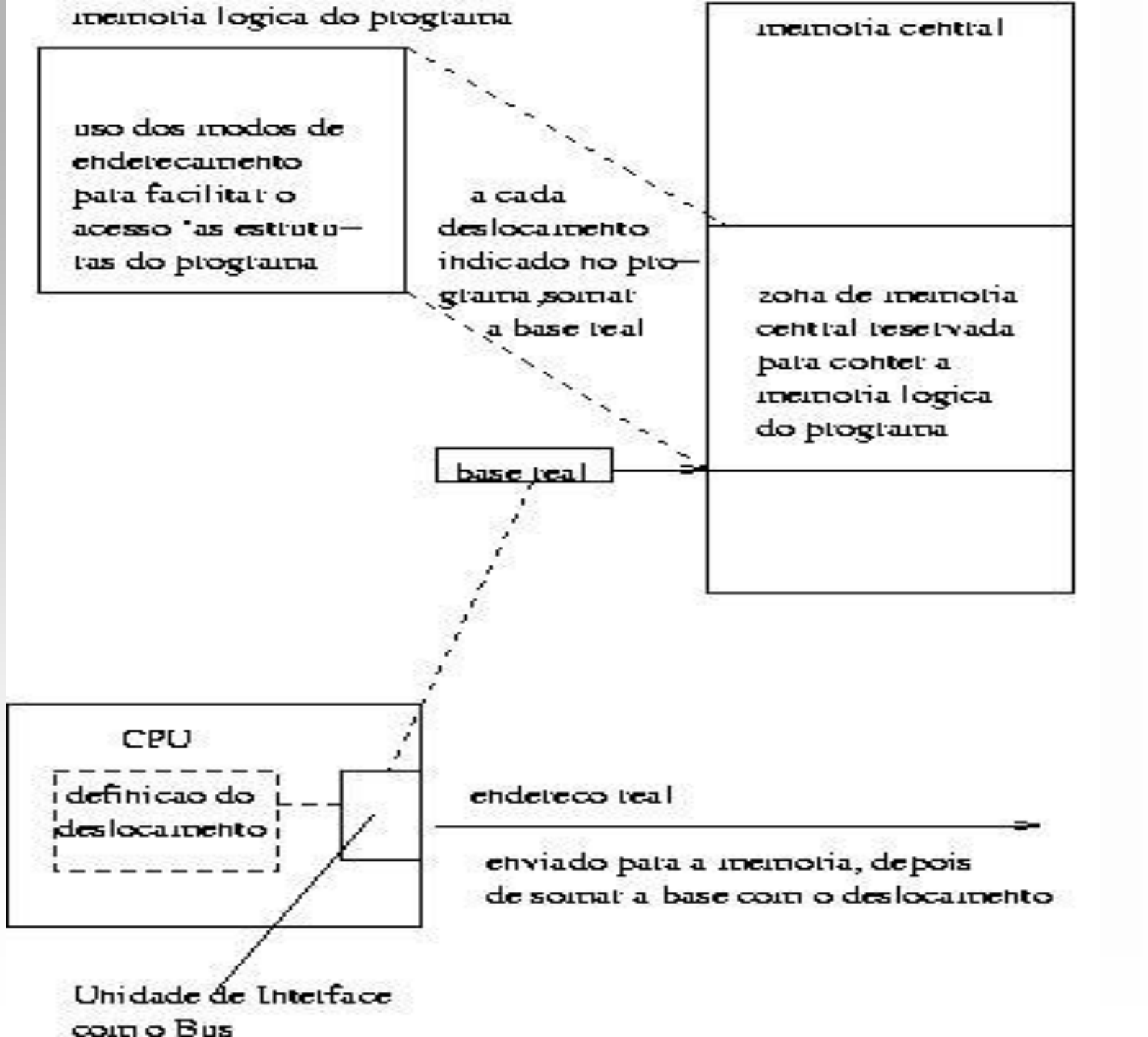
Exemplo:

- Operandos e resultado em memória
- Memória de 64 KiloBytes = 2^{16} Bytes

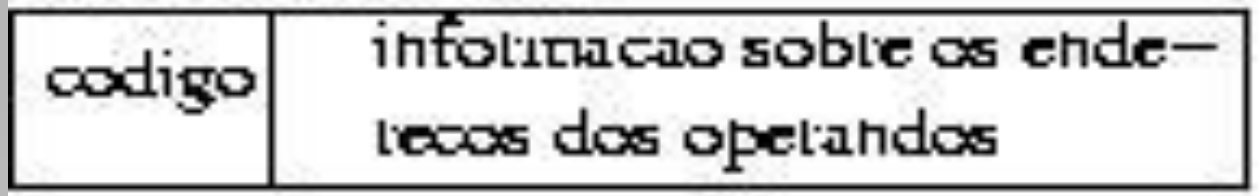
Então: 8 bits + 3 * 16 = 56 bits -- uma instrução

Que alternativas?

- Instruções com menor número de endereços
- Instruções em que só figura uma parte do endereço
- Instruções mais simples terão menos argumentos
- Instruções com operandos e resultado em registadores do CPU: ´´end´´ terá menos bits
- Instruções podem assumir que os operandos estão em células pré-definidas


memoria logica do programa
memoria central
uso dos modos de endereçamento para facilitar o acesso as estruturas do programa
a cada deslocamento indicado no programa somar a base real
zona de memoria central reservada para conter a memoria logica do programa
base real
CPU
definicao do deslocamento
endereco real
enviado para a memoria, depois de somar a base com o deslocamento
Unidade de Interface com o Bus

Instrucao maquina



conforme o modo de endereçamento

produz o endereco efectivo

envia-o para a unidade de interface com o bus externo (Memory Address Register)

Instruções de 3 endereços

código – op1 -- op2 – dest

end1 = mod(op1)

end2 = mod(op2)

end3 = mod(dest)

(end3) ← código((end1),(end2))

Instruções de 2 endereços

código – op1 – dest

end1 = mod(op1)

end2 = mod(dest)

Ou só tem 1 operando, ou tem 2 operandos mas esmaga o segundo:

(end2) ← código((end1),(end2))

Instruções de 1 endereço

código – op

end = mod(op)

Se tiver 2 operandos, assume que um deles foi antes carregado num registador do CPU, implícito (Acumulador AC)

(AC) ← código((end),(AC))

Instruções sem endereços

código

Assumem uma estrutura implícita, na memória ou no processador, sobre a qual operam.

Exemplo: operações sobre uma PILHA (*stack*)

Empilhar Registador: (*push*)

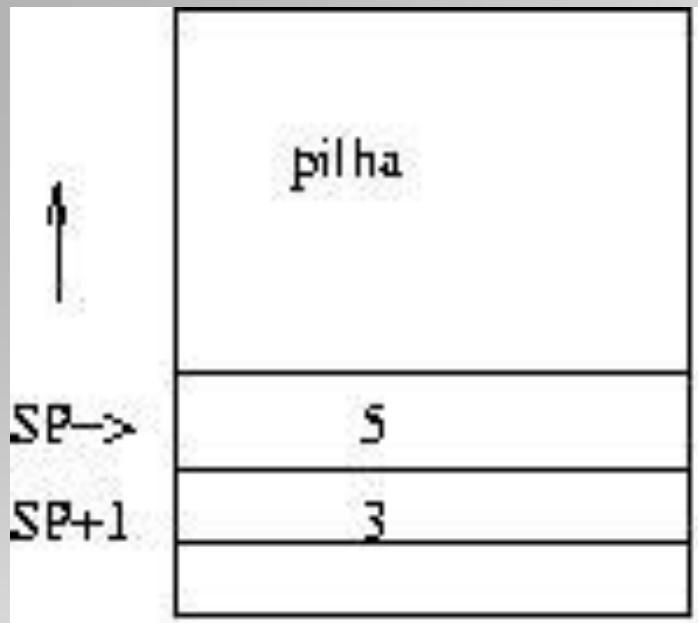
- Cria uma nova posição sobre o Topo da Pilha
- Copia (Registador) para o novo Topo da Pilha

Desempilhar Registador (*pop*)

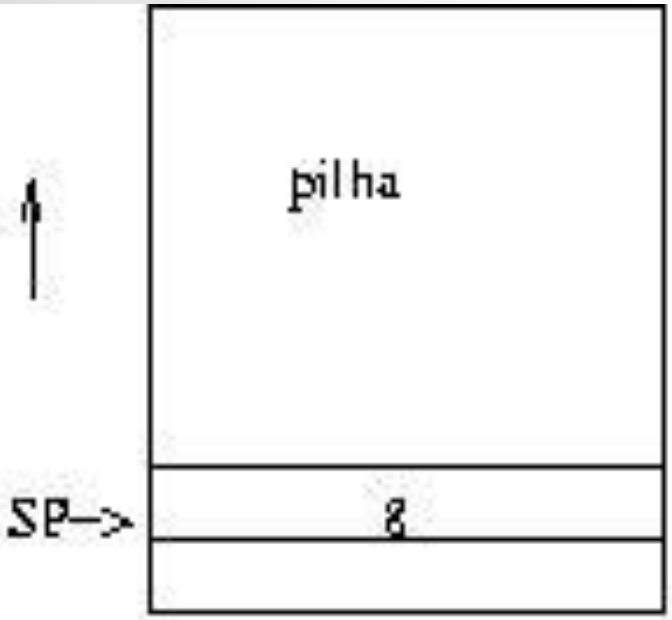
- Copia o valor do Topo da Pilha para o Registador
- e elimina esse valor do Topo da Pilha

Somar

- Desempilha, sucessivamente, os dois elementos no Topo da Pilha e empilha o resultado



(a)



(b)

Empilhar AC:

decrementa SP;

M[SP] := (AC)

sentido decrescente do valor de SP

Desempilhar AC:

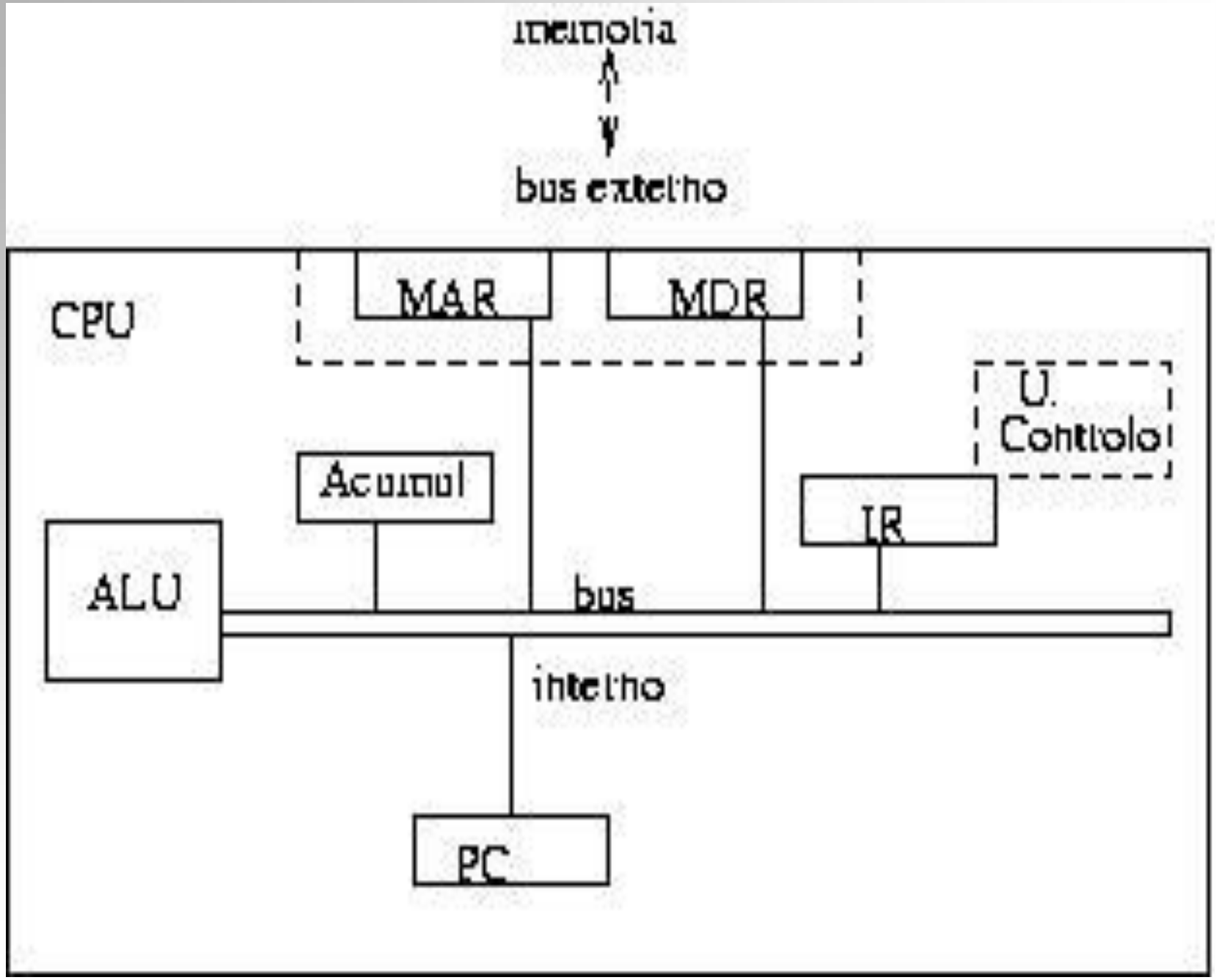
(AC) := M[SP];

incrementa SP;

operacao:

M[SP+1] :=codigo(M[SP], M[SP+1])

Exemplo de um CPU simplificado


memória
bus externo
CPU
MAR
MDR
U. Controlo
Acumul
IR
ALU
bus
interno
PC

Instruções com um formato de 8 bits:

Código I endereço

Código: 3 bits, codifica 8 instruções

Endereço: 5 bits, permite endereçar directamente até 32 posições de memória (bytes)

No CPU:

ALU opera sobre números de 8 bits

PC de 5 bits

IR de 8 bits

Registador de trabalho AC (acumulador) de 8 bits

MAR de 5 bits

MDR de 8 bits

: Na Memória central

Capacidade de 32 células, cada contém 1 byte

Bus de sistema:

5 linhas de endereços

8 linhas de dados

linhas de controlo

Instruções de 1 endereço

load n ----- AC := Mem[n] --- mov AC,n

store n ----- Mem[n] := AC --- mov n, AC

add n ----- AC := AC + Mem[n]

sub n ----- AC := AC – Mem[n]

jump n ----- PC := n

jpos n ----- IF AC > 0 THEN PC := n;

jzero n ----- IF AC == 0 THEN PC := n;

halt ----- Indicador de Paragem := TRUE

em que n é um endereço de memória

Assume-se que:

- Inicialmente PC := 0
- O Programa é carregado em posições consecutivas de memória, a partir do endereço 0

O ciclo do CPU é:

```
WHILE NOT Indicador de Paragem DO
   BEGIN
        IR :=  Mem[PC];   --- FETCH
        PC := PC + 1;       --- prepara próxima I. (!)
        ExecutarInstrução (IR)   --- executa I. corrente
   END;
```

Transferências (´´moves´´)

load n – mov AC, n

carregar AC com o valor de Mem[n]

MAR := n;

Bus de controlo := pedir-leitura

Aguardar;

MDR := valor no Bus de dados

AC := MDR

store n – mov n, AC

carregar o valor de AC em Mem[n]

MAR := n;

MDR := AC

Bus de controlo := pedir-escrita

Instruções Aritméticas

add n -acumular em AC a soma de AC e Mem[n]

MAR := n;

Bus de controlo := pedir-leitura

Aguardar;

MDR := valor no Bus de dados

Activar ALU:

    res := AC + MDR

AC := res

Instruções de salto

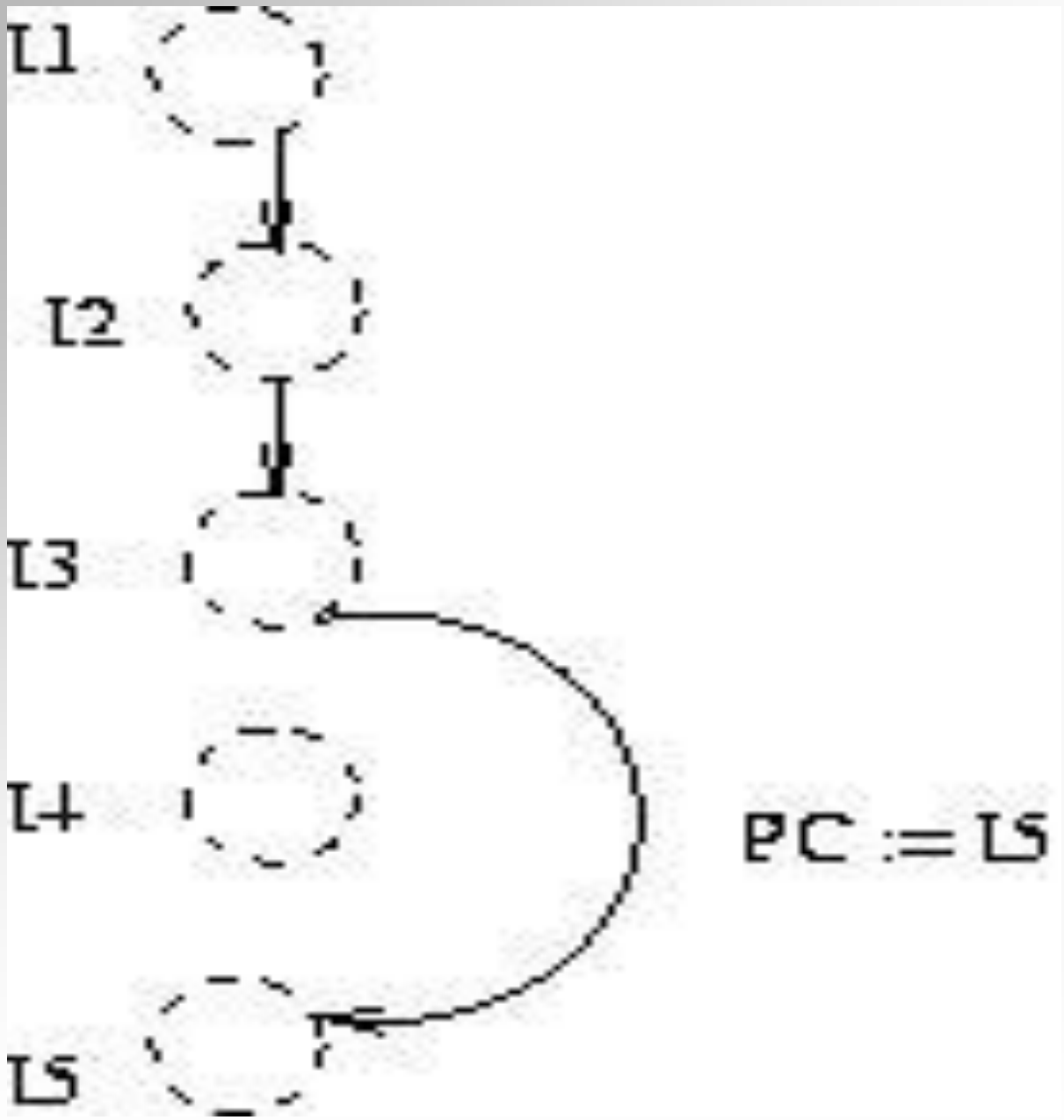
jump n ----- PC := n

jpos n ----- IF AC > 0 THEN PC := n;

jzero n ----- IF AC == 0 THEN PC := n;

Afectam o conteúdo do PC que determina o endereço onde o CPU irá obter a próxima instrução

Se jpos n e jzero n não alterarem o valor do PC, então este valor aponta o endereço seguinte da zona de memória onde está o programa.


L1
L2
L3
L4
L5
PC := L5

Exemplo de programa

Multiplicar os conteúdos de 2 células de memória de endereços de memória E1 e E2 e colocar o resultado na célula de endereço E3.

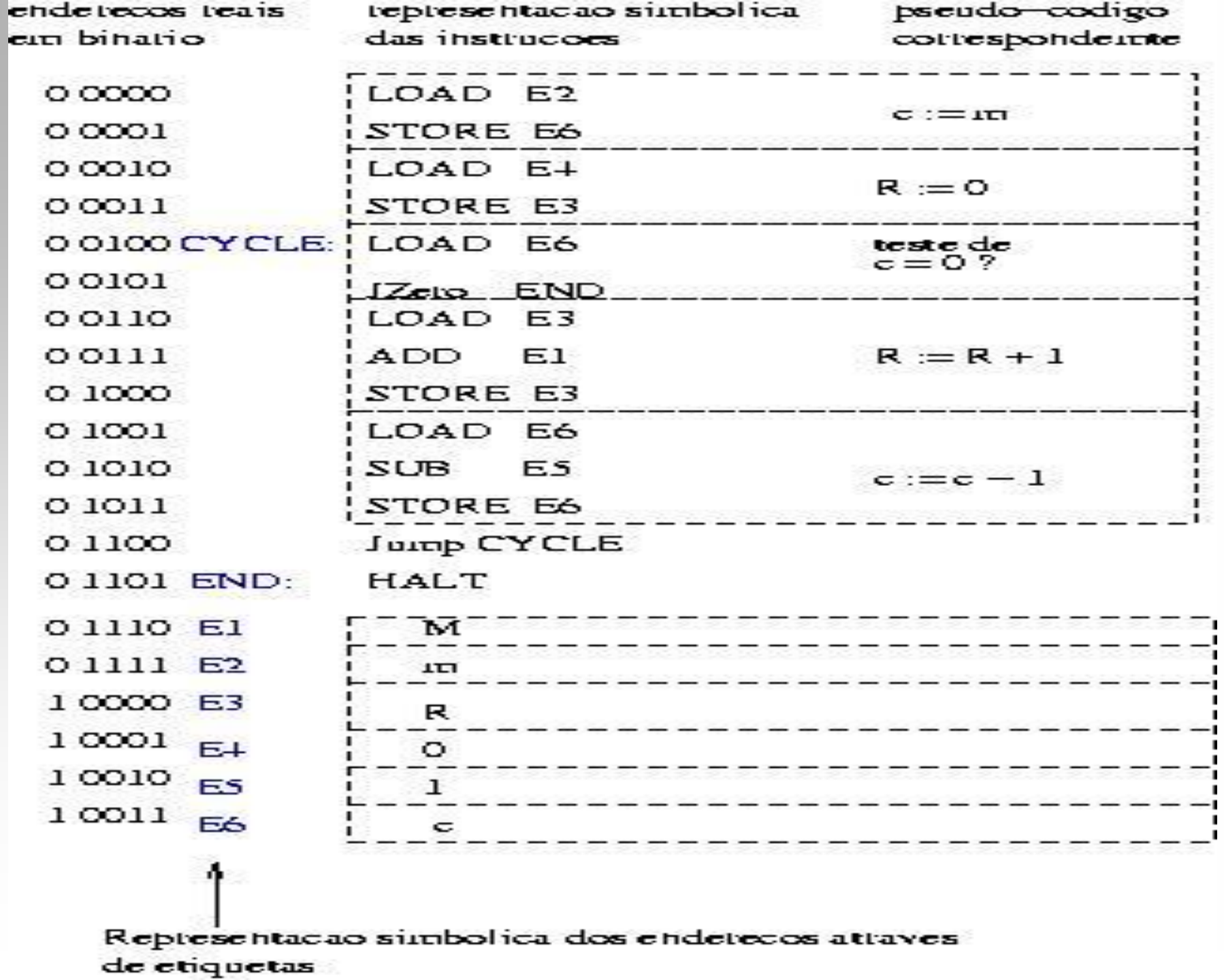
M = Mem[E1]

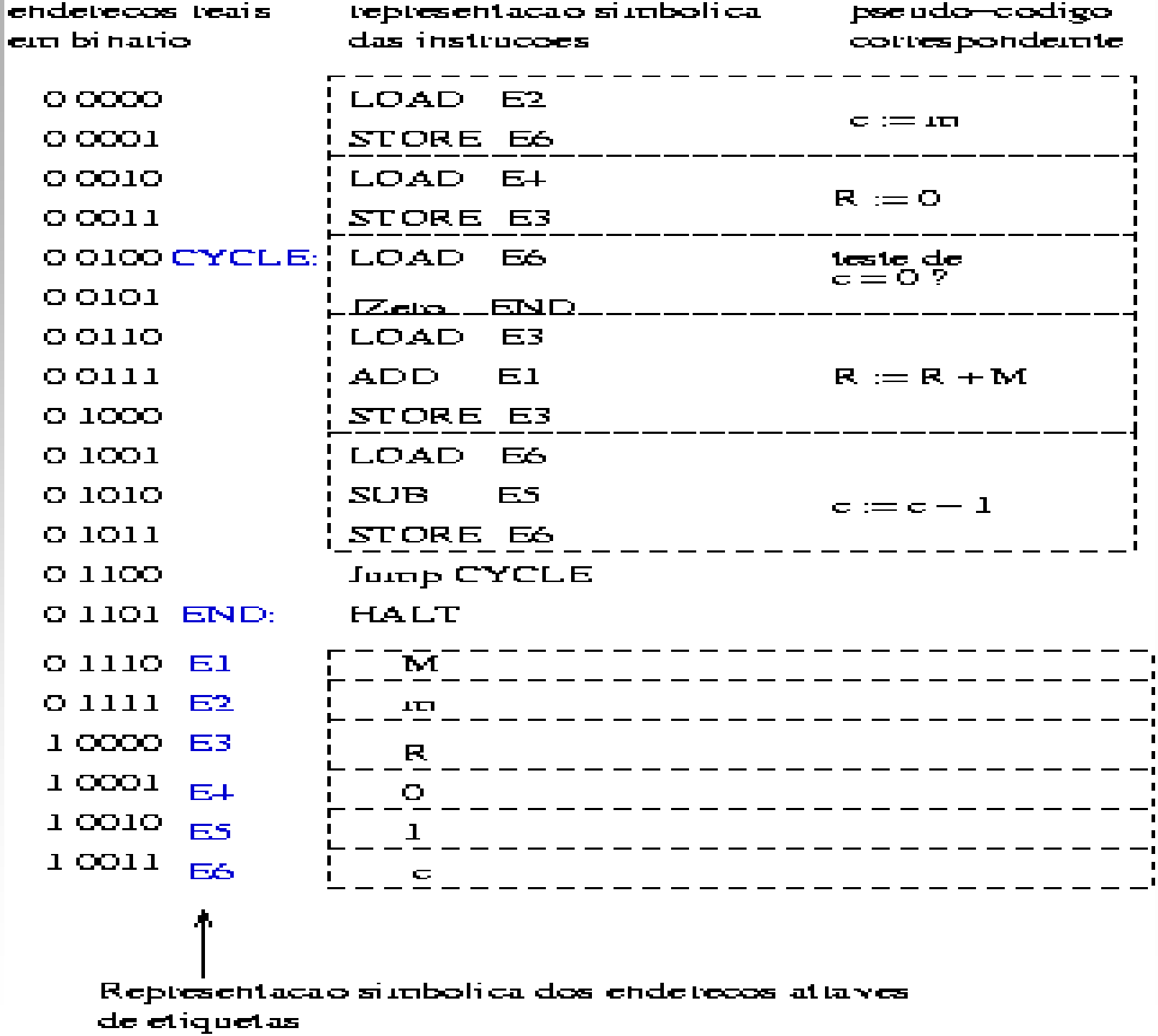
m = Mem[E2] R : = M * m

R = Mem[E3]

Algoritmo baseado em somas sucessivas:

```
c := m; R := 0;
Enquanto c <> 0 fazer
        c := c – 1;
        R := R + M
fimfazer;
```


endetecos teais em binatio
representacao simbolica das instrucoes
pseudo-codigo correspondente
0 0000 LOAD E2
0 0001 STORE E6
c := m
0 0010 LOAD E4
0 0011 STORE E3
R := 0
0 0100 CYCLE: LOAD E6
0 0101 JZero END
teste de c = 0 ?
0 0110 LOAD E3
0 0111 ADD E1
0 1000 STORE E3
R := R + 1
0 1001 LOAD E6
0 1010 SUB E5
0 1011 STORE E6
c := c - 1
0 1100 Jump CYCLE
0 1101 END: HALT
0 1110 E1 M
0 1111 E2 m
1 0000 E3 R
1 0001 E4 0
1 0010 E5 1
1 0011 E6 c
Representacao simbolica dos endetecos atraves de etiquetas


endereços reais em binário
representação simbólica das instruções
pseudo-código correspondente
0 0000
0 0001
LOAD E2
STORE E6
c := m
0 0010
0 0011
LOAD E4
STORE E3
R := 0
0 0100 CYCLE:
0 0101
LOAD E6
Zero END
teste de c = 0 ?
0 0110
0 0111
0 1000
LOAD E3
ADD E1
STORE E3
R := R + M
0 1001
0 1010
0 1011
LOAD E6
SUB E5
STORE E6
c := c - 1
0 1100
Jump CYCLE
0 1101 END:
HALT
0 1110 E1
0 1111 E2
1 0000 E3
1 0001 E4
1 0010 E5
1 0011 E6
M
m
R
0
1
c
Representação simbólica dos endereços através de etiquetas