Departamento de Engenharia Elétrica e de Computação
EESC-USP

SEL-0415 Introdução à Organização de Computadores

Conjunto de Instruções e Modelos de Arquiteturas

Aula 7

Prof. Marcelo Andrade da Costa Vieira
Profa. Luiza Maria Romeiro Codá
Porfa. Maria Stela Veludo de Paiva

Padrões de Códigos de Caracteres

Em teclados, as teclas vem especificadas com os caracteres que utilizamos na nossa comunicação escrita; os monitores de vídeo também apresentam na tela os mesmos tipo de caracteres.

Mas o computador só trabalha com 0s e 1s.

Cada caractere do teclado é convertido para um **padrão de bits**

No monitor de vídeo cada padrão de bits é convertido para o seu caractere correspondente

Padrões mais utilizados para representação de caracteres:

- Código ASCII
- Código EBCDIC
- UNICODE

Microcomputador de 8 bits
Arquitetura de Von Neumann

Exemplo:

Código ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange)

. a cada caracter atribui um código de **7** bits, podendo representar 128 caracteres

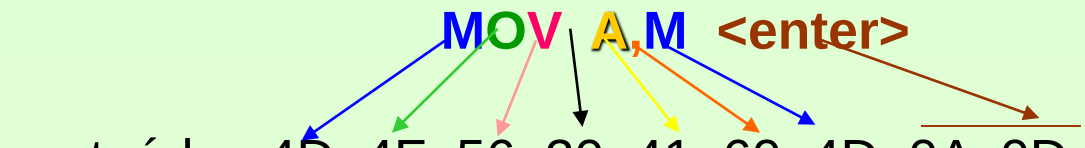
. Conjunto de caracteres: letras, números, sinais de acentuação e pontuação, e caracteres de controle

Exemplo: teclado alfanumérico

. O usuário digita o caracter **A** e o microcomputador recebe o código **41H**

. Conteúdo do **programa fonte**: Código ASCII

Exemplo: arquivo fonte Prog1.asm



CÓDIGO ASCII

*	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

OEM Extended ASCII

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
9	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	¢	£	¥	₧	ƒ
A	á	í	ó	ú	ñ	Ñ	ª	º	¿	⌐	¬	½	¼	¡	«	»
B	░	▒	▓	│	┤	╡	╢	╖	╕	╣	║	╗	╝	╜	╛	┐
C	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧
D	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	█	▄	▌	▐	▀
E	α	ß	Γ	π	Σ	σ	µ	τ	Φ	Θ	Ω	δ	∞	ø	∈	∩
F	≡	±	≥	≤	⌠	⌡	÷	≈	°	∙	·	√	ⁿ	²	■	

cplusplus.com

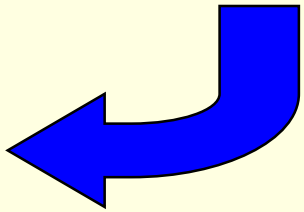
INSTRUÇÕES

- Padrão de código binário armazenado em um dispositivo de memória para comandar o microprocessador na execução de determinada tarefa:
  - *Microcontroladores*: gravado na memória de programa (tipo ROM);
  - *PC*: gravado em arquivo *.exe, que é carregado na memória principal (RAM) para ser executado;
- Cada máquina tem seu próprio **conjunto de instruções** baseadas no projeto de seu µP;

Conjunto de Instruções - Programa

- **Fabricante:** seleciona combinações de padrões de bits (***opcodes***), atribuindo-lhes um significado, utilizando circuitos lógicos específicos

Conjunto de Instruções



- **número de bits que a ULA reconhece e processa de uma só vez define o tipo de microprocessador**

- Usualmente de *4 bits para os mais simples até de 64 bits para computadores de alta velocidade* -

Conjunto de Instruções - Programa

- Cada instrução tem a ela associado um **código binário de operação** (**OPCODE**)
- Além do **opcode** a instrução pode conter mais bytes que representam **dados** ou **endereços associados à instrução**;
- O tamanho do **OPCODE** varia de acordo com a arquitetura do µP;
  - **Ex.:** máquina com OPCODE de 8 bits pode ter até 256 (2^8) tipos de instruções diferentes

Uma instrução é constituída pelo seu **opcode,** podendo incluir também bytes extras, que representam **dados ou endereços** associados à instrução

Conjunto de Instruções - Programa

- O **OPCODE** especifica :
  - a função da instrução
  - os registradores usados pela instrução (se houverem)
  - se existem mais bytes na memória de programa, pertencentes a este **opcode**, que precisam ser buscados (lidos)

Conjunto de Instruções - Programa

- **Programa em Linguagem de Máquina** ➡ programa escrito com **Instruções Binárias**
  - Ex.: 00100100 (opcode)

    00010011 (dado de 8 bits)

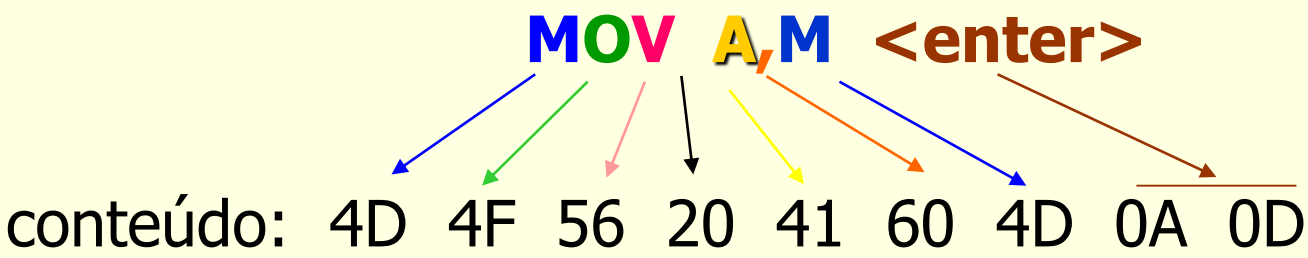
    significa **Soma: A = A + 13H**

- Para o homem, os programas em linguagem de máquina são difíceis de serem escritos e não ilustram as operações

Conjunto de Instruções - Programa

- **Programa Fonte ➡** programa de computador em sua forma original, anotado pelo programador em uma linguagem-fonte, antes de ser codificado em instruções de linguagem de máquina. Seu conteúdo é o caracter do mnemônico em Código de representação de caractere(ASCII)

Exemplo: arquivo fonte Prog1.asm



Conjunto de Instruções - Programa

- Os fabricantes criaram palavras curtas em inglês que representam a instrução binária na máquina (*MNEMÔNICOS*), *facilitando a escrita dos programas.*
- **Programa em Linguagem Assembly ➡** programa escrito utilizando *MNEMÔNICOS.*

- **EX: ADD A,#13H ; soma conteúdo de A com o valor 13H**

Conjunto de Instruções - Programa

- **Programas em Linguagem Assembly** necessitam de um programa tradutor (Assembler) : converte o programa para Linguagem de máquina.

- **Programas em Linguagens de alto nível** (Fortran, Pascal, C, Basic...) necessitam de um programa tradutor (**compilador**) para transformar em linguagem de máquina.

Conjunto de Instruções - Programa

- Cada linha de um programa em **Linguagem Assembly** é constituída por quatro campos distintos:
  - **rótulo (Label):** nome atribuído ao endereço da instrução
  - **Opcode**
  - **Operandos:** podem ser registradores e/ou valores que representam dados ou endereços da instrução
  - **comentários :** precedidos por ;

exemplo:

```
Label    Código de Operação   Operando    Comentário

saída:          MOV              A, #80H    ; move imediato 80H para A
```

Conjunto de Instruções - Programa

- Um programa em **Linguagem Assembly** inclui instruções especiais **(pseudo-instruções ou Diretivas),** que são usadas pelo programa tradutor para a **estruturação do programa.**
- **Pseudo-instruções não tem opcode**

- **Exemplo:**

1 - ORG endereço

- **Função do ORG:** o endereço especificado no operando representa endereço inicial da área de programa, ou subrotina ou área de dados (Ex: **ORG 1000H** (O endereço da próxima instrução é 1000H)

2 - Nome EQU valor

- **Função da diretiva EQU:** O **Nome** especificado é associado com o **valor** especificado no operando. (Ex: **val1 EQU 89H**)

Conjunto de Instruções - Programa

- **Exemplo de Diretivas (cont):**

3- DB dado1(8bits), dado2(8bits),.....dadoN(8bits)

- **Função do DB:** os dados de oito bits do operando são inseridos em sequência, diretamente no programa objeto.
- (EX: Dados: DB 01H, 2AH. 3CH)

4 - END

- **Função :** Indica até onde o programa deve ser compilado.

Programa do uC Atmel 8051

Exemplo de um Programa Assembly do 8051

Mnemônicos (Programa Assembly)

```
        ORG 0
LOOP:   MOV A,30H
        CJNE A,#00,LOOP
AQUI:   SJMP AQUI
        END
```

Código Compilado (Opcode)

Addr	Opcodes	ASC	Label	Disassembly
0000	E5 30	å0	LOOP	MOV A,30h
0002	B4 00 FB	´û		CJNE A,#00h,LOOP
0005	80 FE	€þ	AQUI	SJMP AQUI

Programa do uC Atmel 8051

Exemplo de um Programa Assembly do 8051

Memória

00	E5
01	30
02	B4
03	00
04	FB
05	80
06	FE

Endereço Conteúdo

Addr	Opcodes	ASC	Label	Disassembly
0000	E5 30	å0	LOOP	MOV A,30h
0002	B4 00 FB	´ú		CJNE A,#00h,LOOP
0005	80 FE	€þ	AQUI	SJMP AQUI

Conjunto de Instruções - Programa

Programa em Linguagem Assembly (resumo):

- cada instrução do **uP** tem um **código binário** **(opcode)** a ela associado, que especifica a função da instrução e seus operandos.
- os fabricantes criaram **Mnemônicos** (abreviaturas associadas com a função da instrução) para cada instrução, com o objetivo de facilitar a escrita de programas. Exemplo:

  ADD A, #13H ; soma o acumulador com o valor 13H

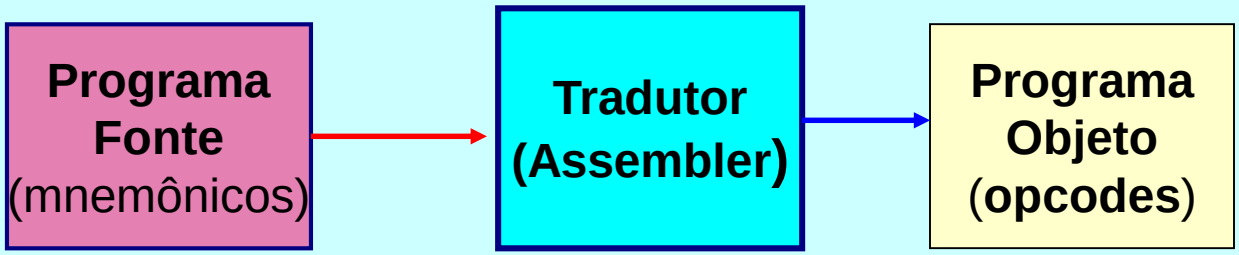
  Mnemônico : **ADD** **Operando: A** e o valor 13H

  o opcode é: 24H e em binário; 00100100
- Programa em **Linguagem Assembly** é um programa contendo mnemônicos.
- Cada **uP** tem seu próprio conjunto de instruções ➡ um programa em Linguagem Assembly é específico para um determinado processador

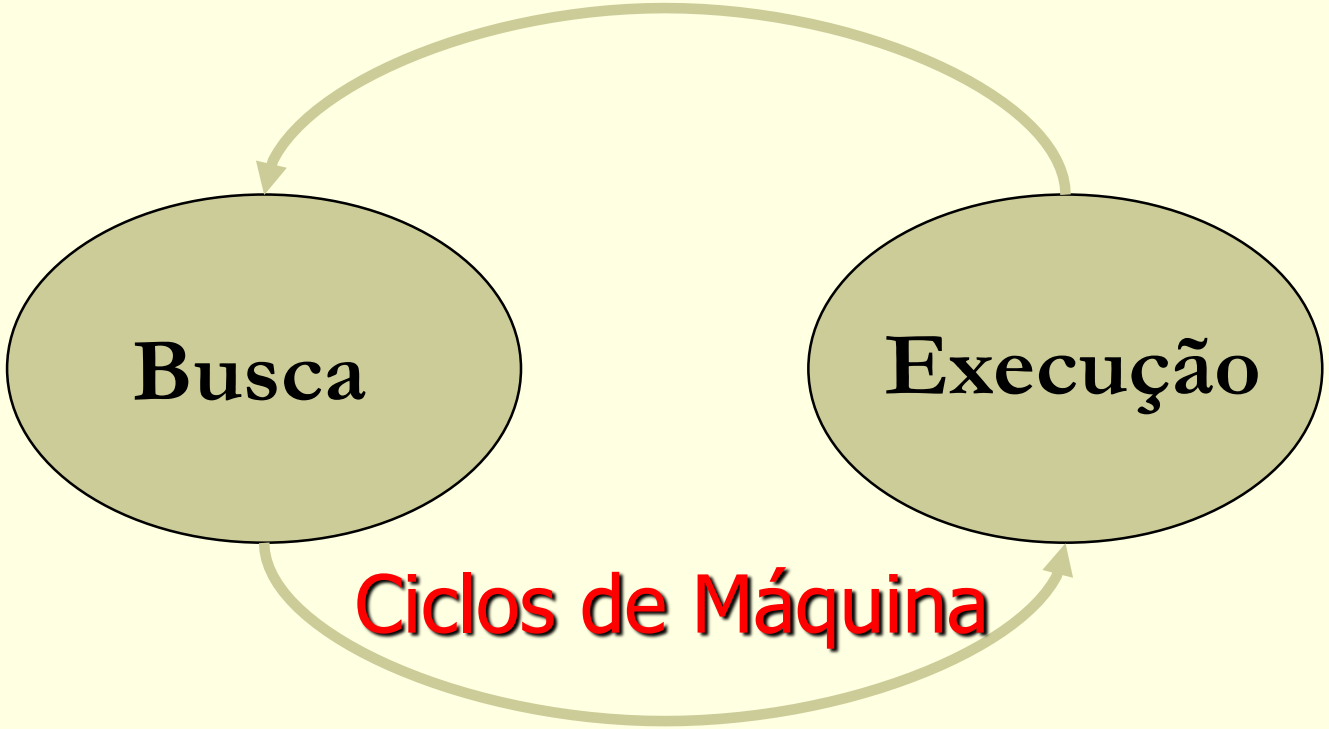
Conjunto de Instruções - Programa

Como o Programa é inserido na memória ROM:

- O programador edita um programa em Linguagem Assembly (**Programa Fonte**), contendo instruções **(Mnemônicos)** do **μP**
- O programador usa um programa tradutor **(Assembler)** para gerar o programa executável (**Programa Objeto**)
- O programa objeto é carregado na ROM através de um circuito **programador** (PROM, EPROM, EEPROM, FLASH)
- A memória ROM, após ser programada, é inserida no circuito do microcomputador



Ciclo de Máquina x Ciclo de Instrução



Cada instrução gasta um ou mais **ciclos de máquina** para ser executada

Ciclo de Máquina x Ciclo de Instrução

Exemplo para o 8051(INTEL):

- Cada ciclo de máquina: ciclo de busca do "opcode" **+** leitura ou gravação, em memória ou I/O (duração de **12T**);
- Existem instruções de 1, 2 e 4 ciclos de máquina (até 48T);

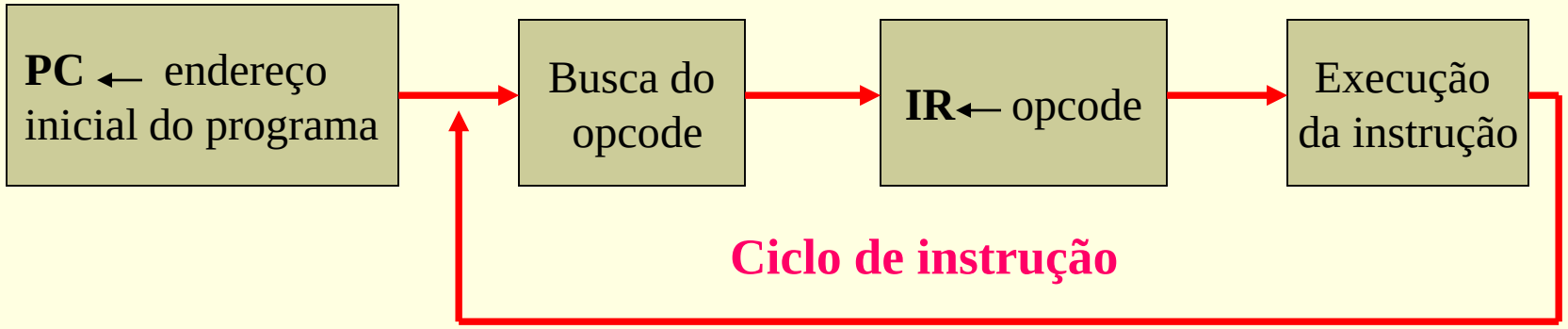
Exemplo para o PIC ("Peripheral Interface Controller") fabricado pela Microchip Technology**:**

- Cada ciclo de máquina: ciclo de busca do "opcode" **+** leitura ou gravação, em memória ou I/O (duração de **4T**);
- Existem instruções de 1 ou 2 ciclos de máquina;

Ciclo de Instrução

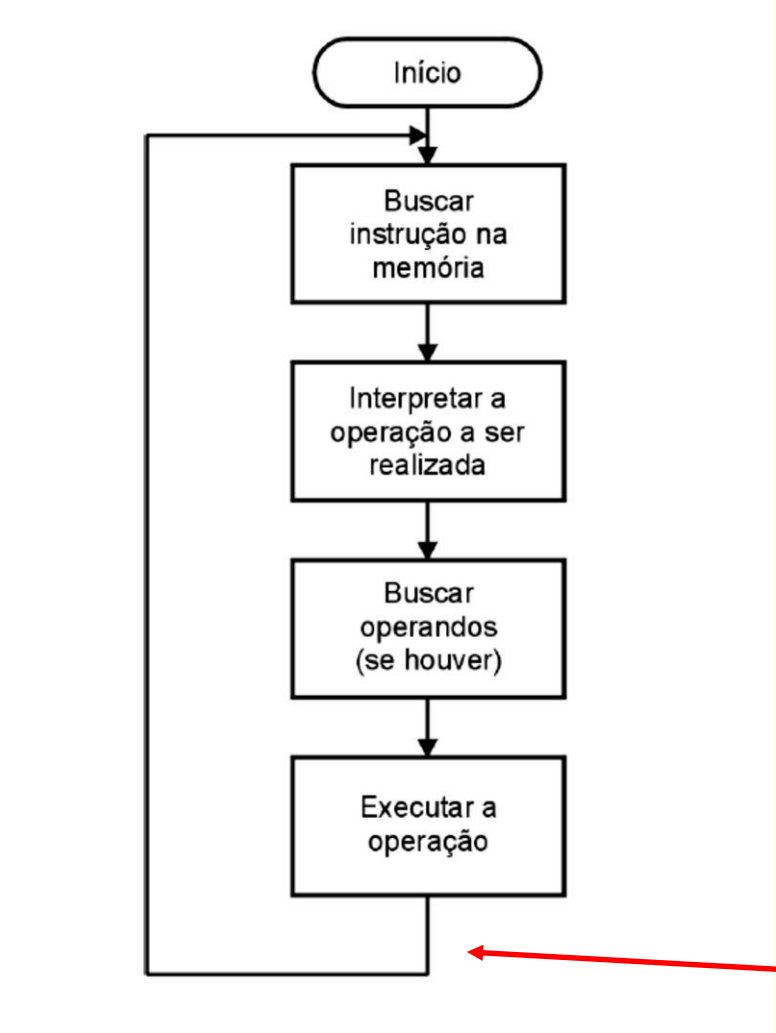
• **Ciclo de Busca:** busca do opcode na memória de programa e armazena no IR;

• **Ciclo de Execução:** executa a instrução armazenada no IR ou busca mais bytes da mesma instrução, se houver, e executa a instrução



O microprocessador executa operações cíclicas

Ciclo de Instrução



Existem **ciclos de instrução** que utilizam mais de **1 ciclo de máquina:**

Não tem "Término"

Ciclo de instrução

Cada instrução é caracterizada por :

- Opcode
- Número de ciclos de máquina
- Número de períodos de clock (T)
- Flags alterados pela instrução
- Modos de endereçamento

Ciclo de Instrução:

CICLO DE **BUSCA DO OPCODE** E SUA **EXECUÇÃO**

Exemplo- Características de uma instrução

Exemplo de um Programa Assembly do 8051

Cada ciclo de máquina gasta 12 períodos de clock (12T

Memória

00	E5
01	30
02	B4
03	00
04	FB
05	80
06	FE

Endereço Conteúdo

Addr	Opcodes	ASC	Label	Disassembly
0000	E5 30	å0	LOOP	MOV A,30h
0002	B4 00 FB	´ù		CJNE A,#00h,LOOP
0005	80 FE	€þ	AQUI	SJMP AQUI

Característica da instrução MOV A,30H:
Número de bytes: 2
Ciclos de máquina : 1
Tempo de execução: 12 T
Se **Freq= 12 MHz** , T = 1/12 us
Portanto, o tempo de execução é = 1us

Classificação da arquitetura quanto ao conjunto de Instruções

Conjunto de Instruções - Processador Específico

CISC - *Complex Instruction Set Computers*

- *Maioria dos μP*
- *Suporta um conjunto maior de instruções*
- *Quantidade de instruções num programa geralmente é menor (mais instruções disponíveis ao programador)*
- *Processamento mais lento (instruções mais complexas que gastam mais de um ciclo de máquina)*

Classificação da arquitetura quanto ao conjunto de Instruções

Conjunto de Instruções - Processador Específico

RISC - *Reduced Instruction Set Computers*

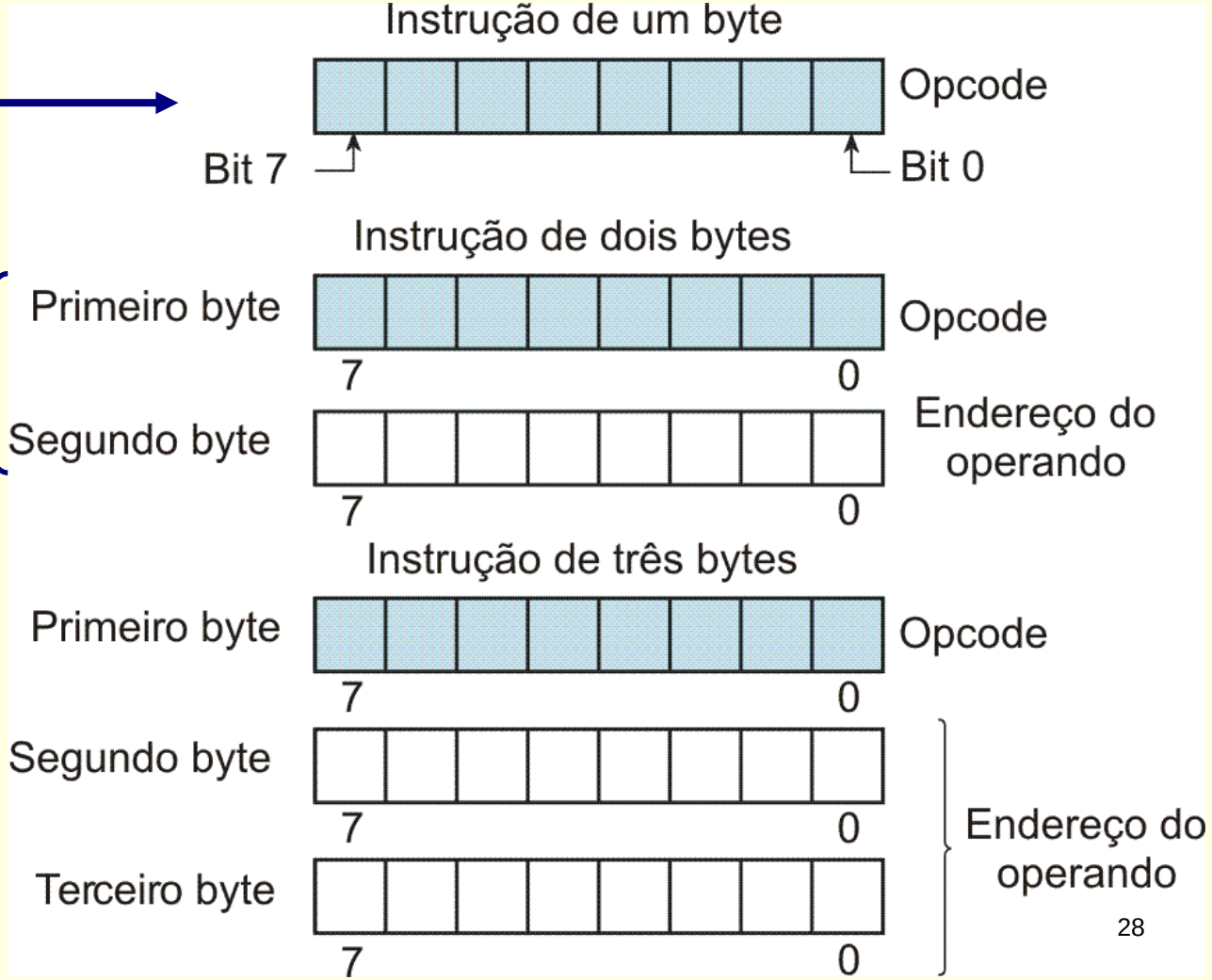
- *Implementa quantidade limitada de instruções ⇨ otimizadas ⇨ maior rapidez – mais simples*
- *Gastam em sua maioria apenas um ciclo de máquina*
- *Todas as instruções ocupam o mesmo tamanho na memória*
- *Programa ⇨ quantidade maior de instruções ⇨ menos instruções disponíveis ao programador.*

RISC x CISC

- **µP CISC**
  - As instruções NÃO necessariamente têm o mesmo comprimento (em bytes).
  - O tempo de execução das instruções é **diferente**, e dependente da freqüência do clock interno do µP.
  - **Mais** instruções disponíveis= programa mais simples
  - Ex: Microcontrolador 8051(Intel)
- **µP RISC**
  - As instruções têm o **mesmo** comprimento (em bytes);
  - O tempo de execução das instruções é **o mesmo** (com exceção para instruções de salto) e são dependentes da freqüência do clock interno do µP.
  - **menos** instruções disponíveis = programa mais complexo
  - Ex: Microcontrolador PIC (Microchip)

Exemplos de Instruções CISC

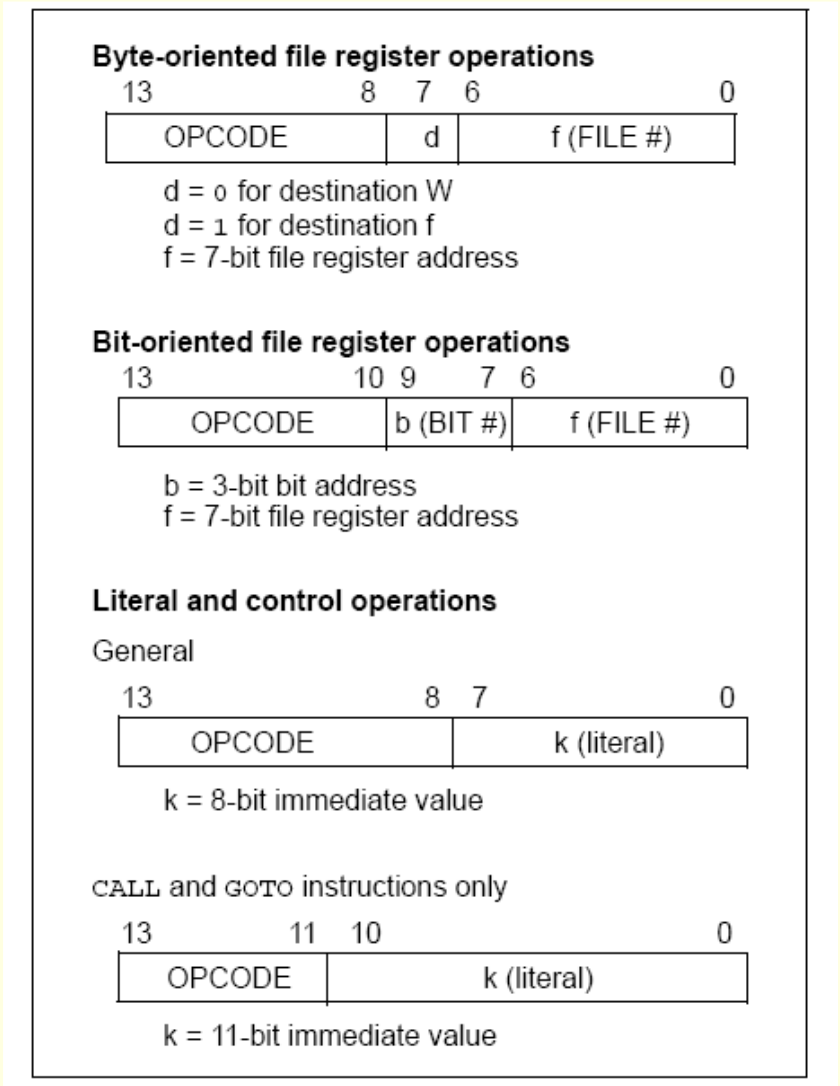
CLR A



MOV A,30h

LJMP 3FB2h

Exemplos de Instruções RISC



MOVF STATUS, W

BCF STATUS, RP0

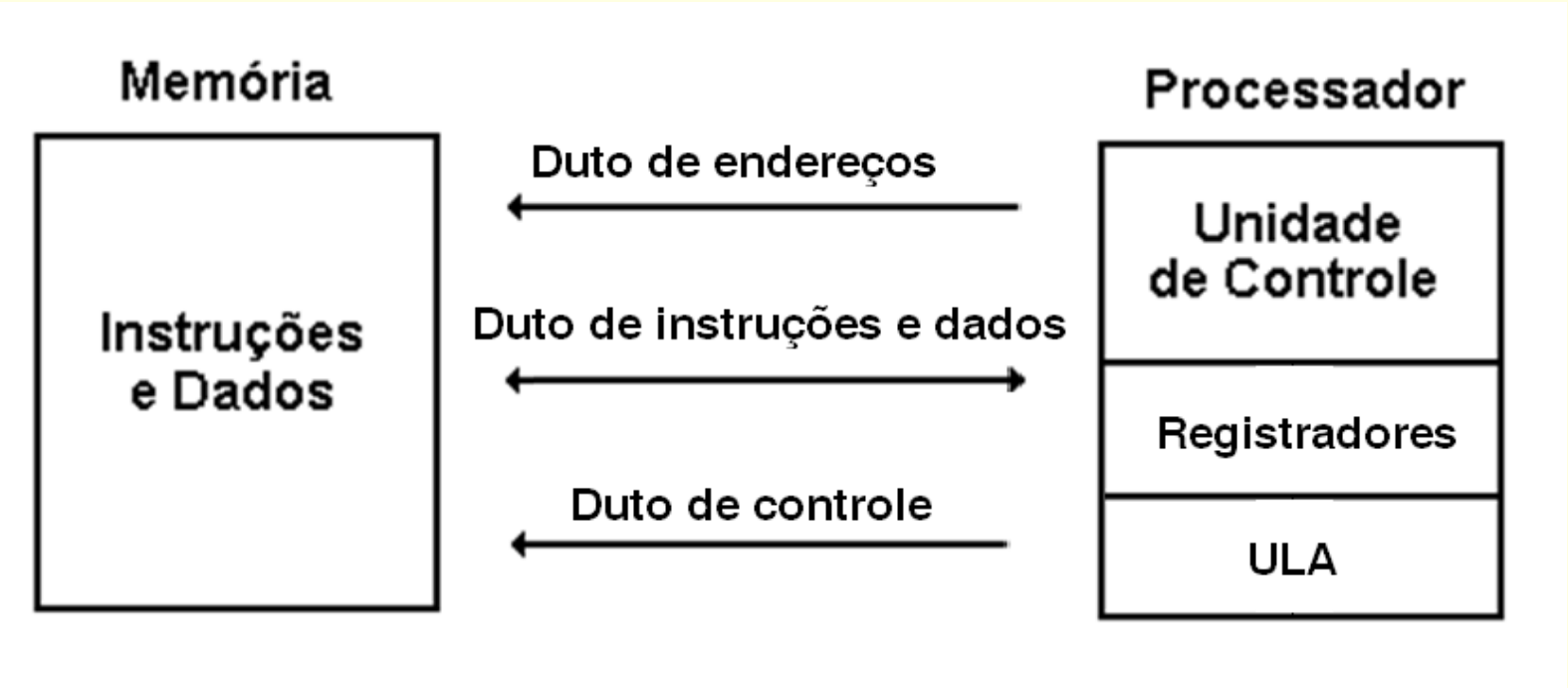
MOVLW B‘00011100’

CALL SUBROTINA

Modelos de Arquiteturas para Computadores

Arquitetura Von Neumann
X
Arquitetura Harvard

Arquitetura Von Neumann



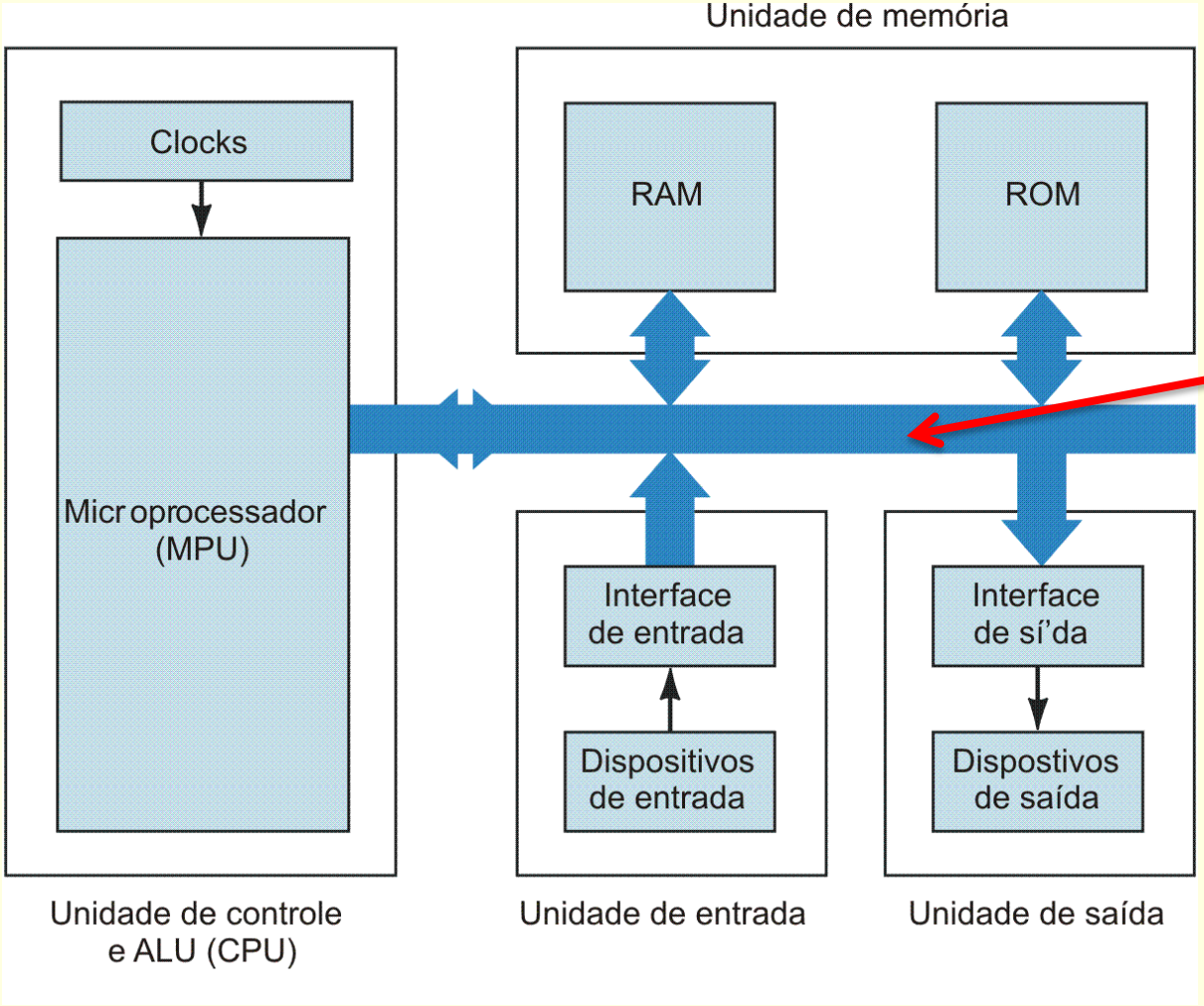
Arquitetura Von Neumann

- **Von Neumann**:
  - Arquitetura mais simples;
  - Mais lento pois não permite acesso simultâneo às memórias;
  - Geralmente CISC

Exemplo:

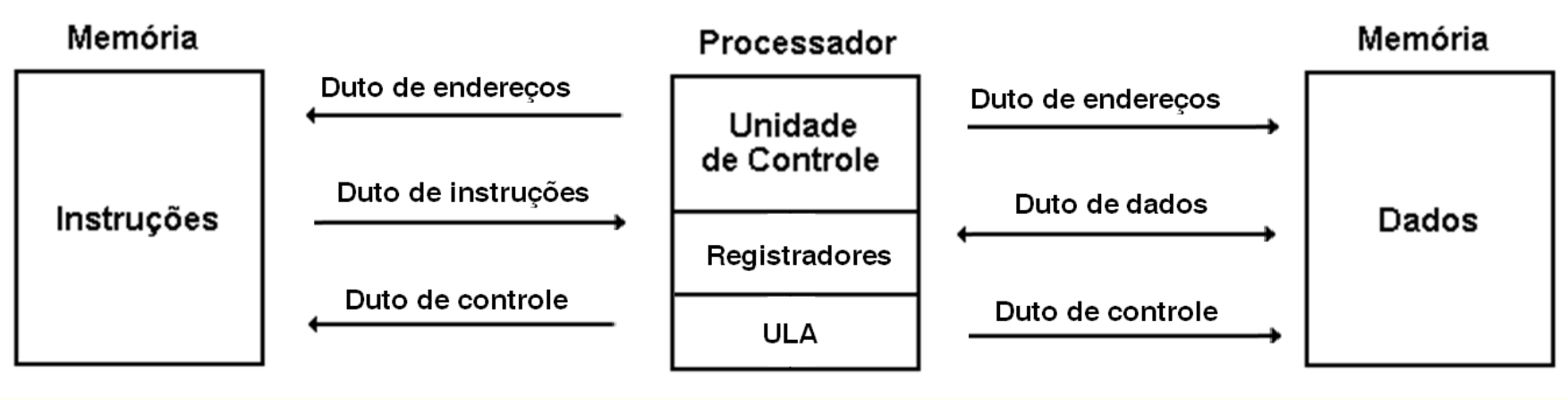
4004	– 46 instruções
8080	– 78 instruções
8085	– 150 instruções
Z80	– Mais de 500 instruções

Microcontrolador Intel 8051 – Arquitetura Von Neumann Modificada (MISTA)



Apesar de duas memórias, elas compartilham o mesmo barramento

Arquitetura Harvard



Arquitetura Harvard

- Busca e execução em apenas 1 ciclo de máquina;
- Todas as instruções tem largura fixa;
- Poucas instruções gastam mais de 1 ciclo de máquina;

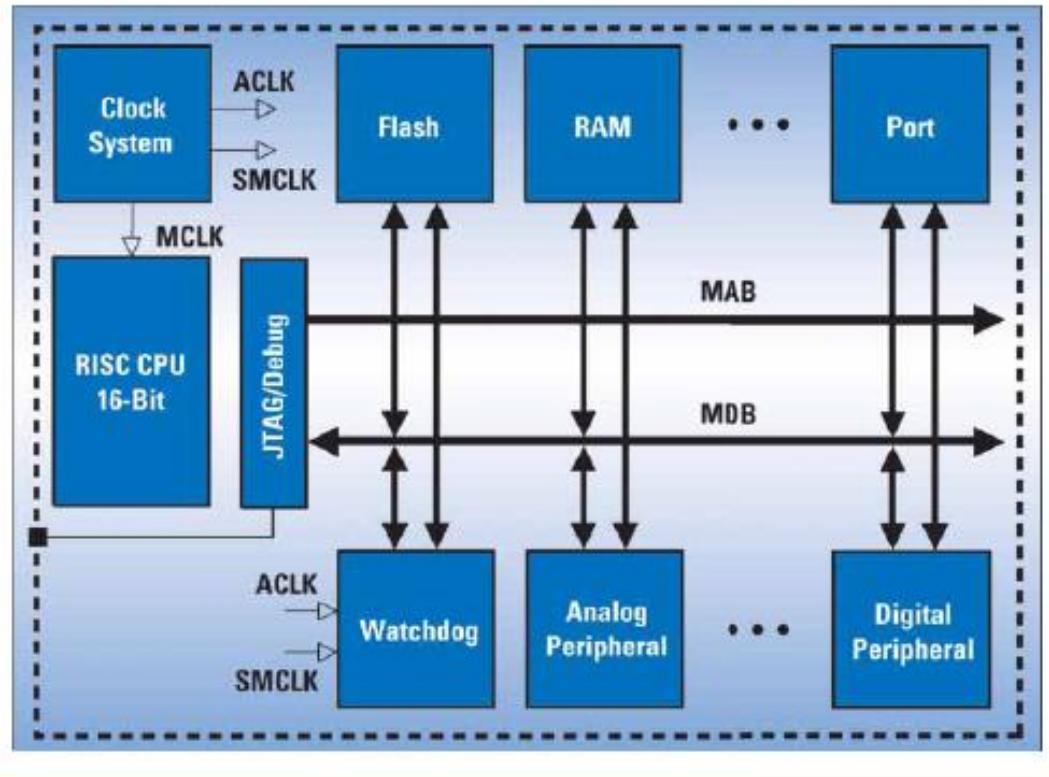
Arquitetura Harvard

- **Harvard:**
  - Arquitetura mais complexa;
  - Mais rápida, pois permite acesso simultâneo às memórias;
  - Geralmente RISC
  - Permite o Pipelining

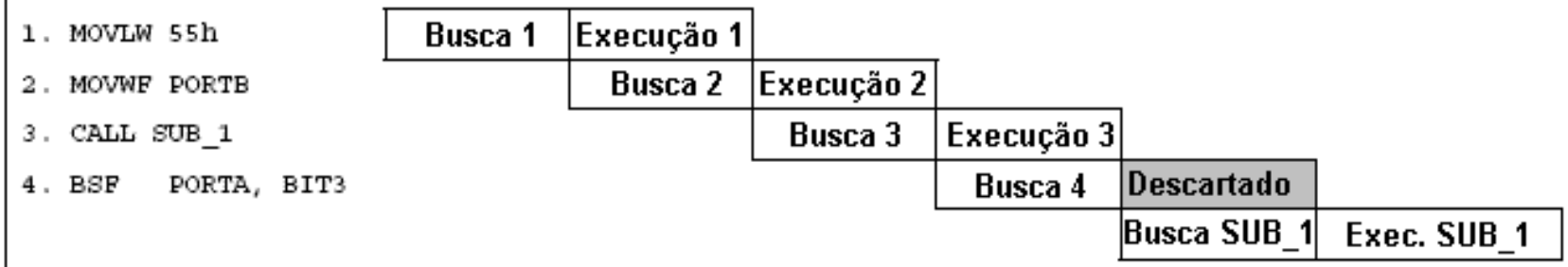
  Exemplo:
  - Processadores Digitais de Sinais (DSP)
  - Microchip PIC – 35 instruções

Arquitetura Von Neummann com Conjunto de Instruções RISC

- **Texas MSP430:**
  - Arquitetura Von Neumann;
  - Instruções RISC de 16 bits;



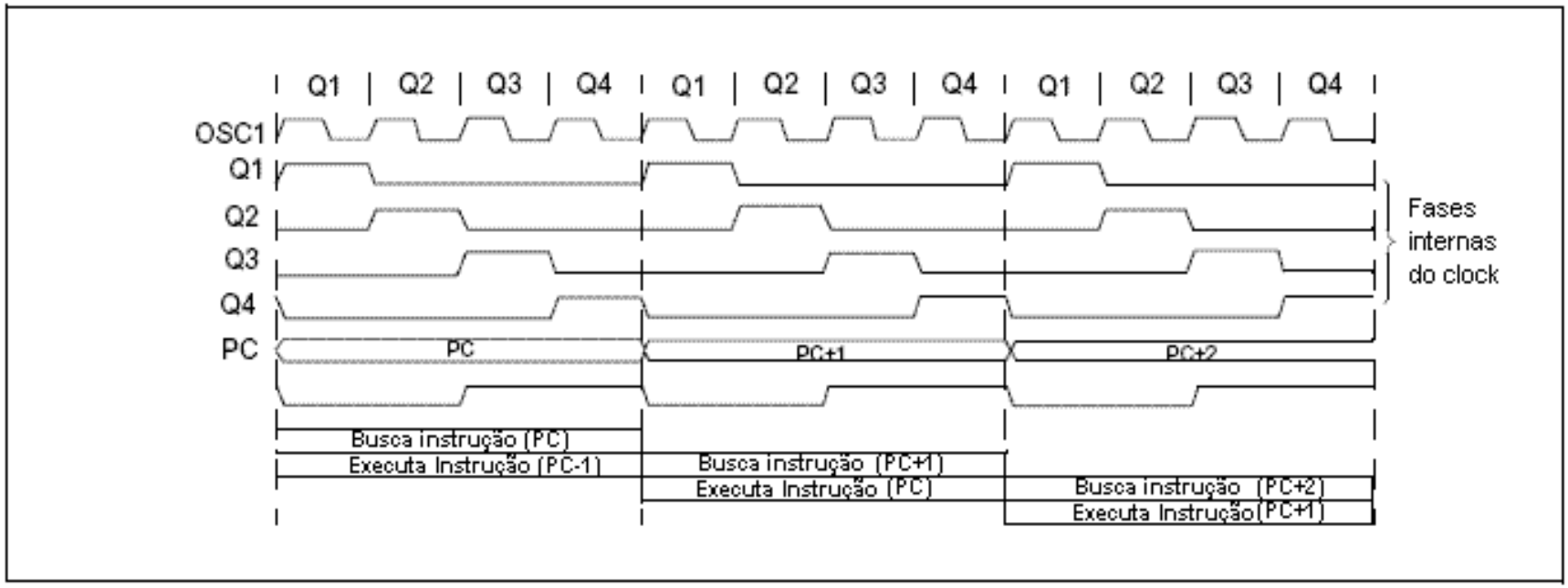
Conceito de Pipeline de 2 estágios: µcontroladores PIC



Todas as instruções "gastam" apenas um ciclo de máquina, exceto para intruções que provocam saltos, que "gastam" dois ciclos de máquina pois precisam esperar que o endereço da próxima instrução seja colocado no pipeline para ser executada.

- No pipeline de instruções do PIC, a busca e execução de instruções é separada em dois estágios , permitindo a busca da próxima instrução, enquanto a instrução atual está sendo executada
- Busca e execução em 1 ciclo de máquina;
- Instruções de “salto” gastam dois ciclos de máquina;

Pipelining de 2 estágios: µcontroladores PIC



- Ciclo de máquina = fosc/4 para os µcontroladores PIC
- As instruções devem ser de um ciclo de máquina

Pipelining

- Pipeline é uma técnica de hardware que permite que a CPU realize a busca de uma ou mais instruções além da próxima a ser executada
- Pipeline é mais fácil de ser implementado em processadores RISC
- Pipelining de 2 ou mais estágios:
  - Intel 8086, 8088, PIC: 2 estágios
  - Pentium: 5 estágios
  - Pentium 4: 20 estágios (ultimo modelo 31 estágios)

FIM