MICROPROCESSADORES E MICROCONTROLADORES

Parte 1

José Wilson Lima Nerys

www.eee.ufg.br/~jwilson

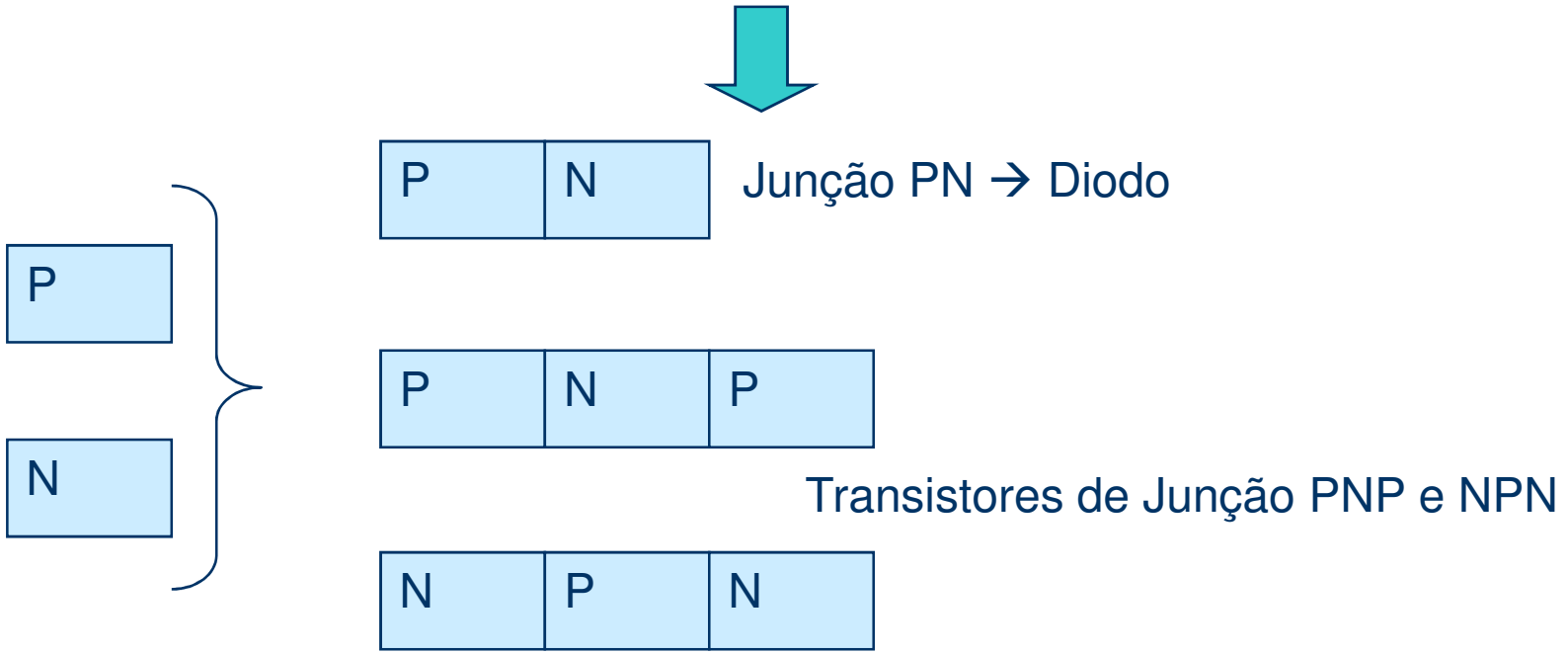
jwilson@eee.ufg.br

MICROPROCESSADORES E MICROCONTROLADORES

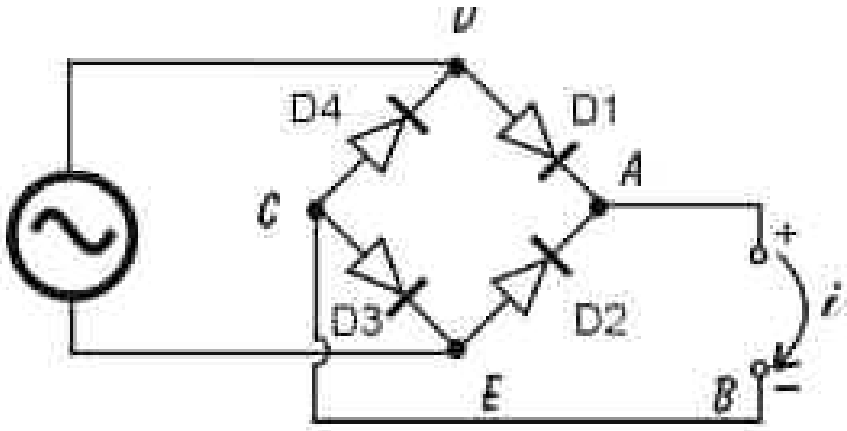
Conceitos Básicos e Princípios de Microprocessadores 8085 e 8086/8088 e Microcontrolador 8051

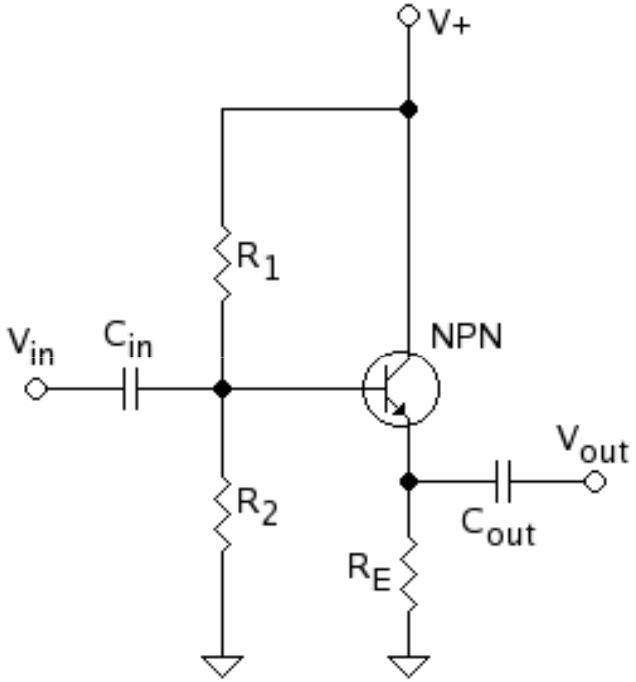
Evolução de Conhecimentos até Microprocessadores

Materiais Elétricos – Estudo de materiais isolantes, condutores e semicondutores – características.

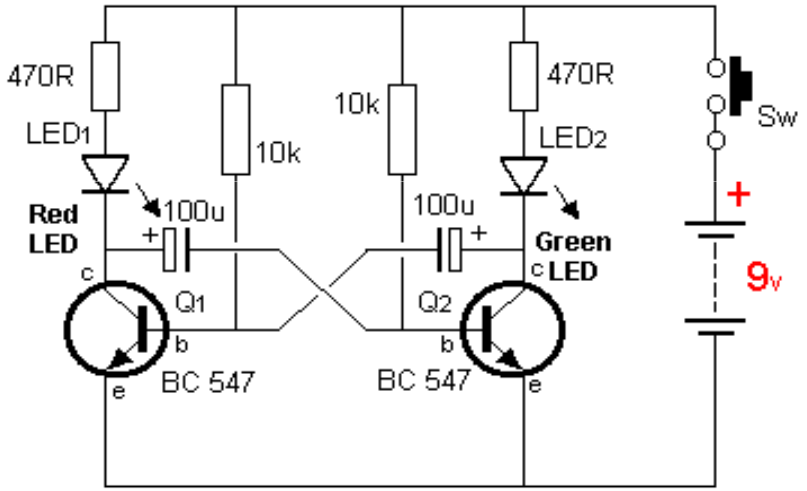


Eletrônica – Aplicações de Diodos e Transistores – retificadores, amplificadores, ceifadores, filtros, multivibradores biestáveis.

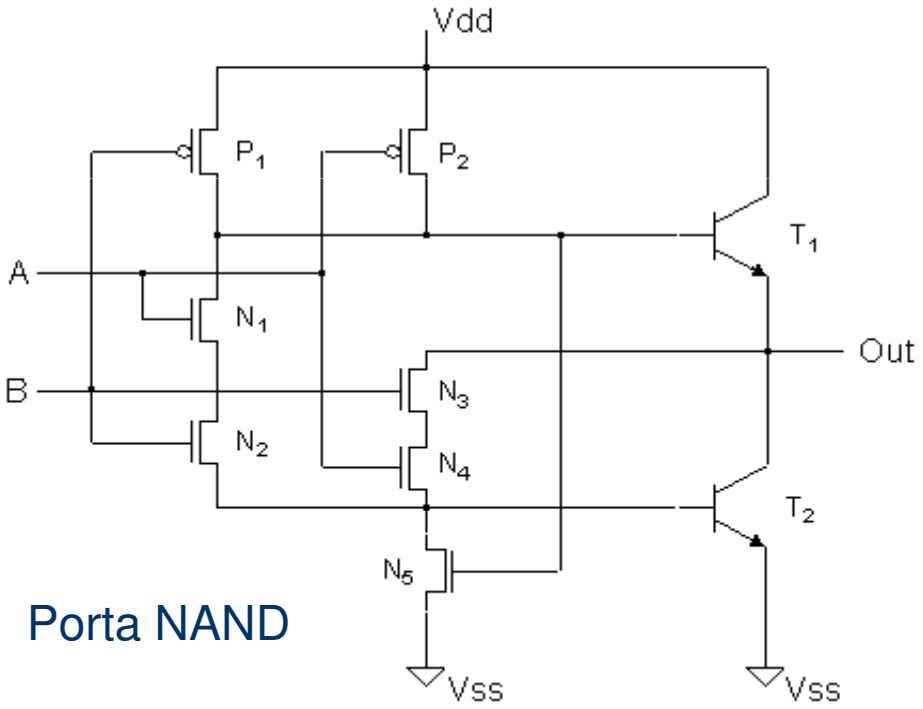




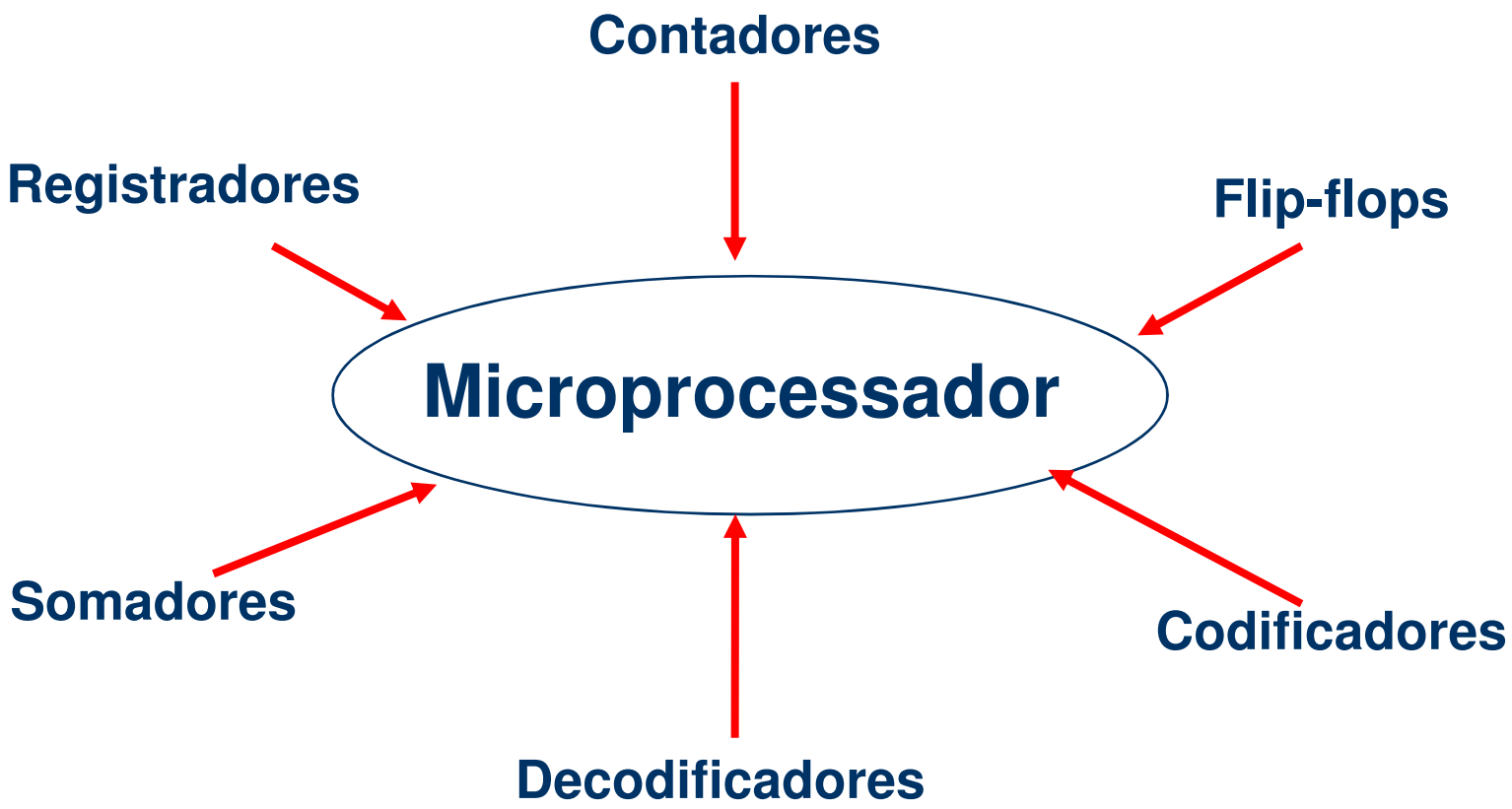
Sistemas Digitais – Sistemas de numeração e códigos binários. Portas Lógicas. Álgebra Booleana. Circuitos lógicos combinacionais. Codificadores, decodificadores, multiplexadores e demultiplexadores. Aritmética binária. Circuitos lógicos seqüenciais (contadores e registradores).



Flip-flop

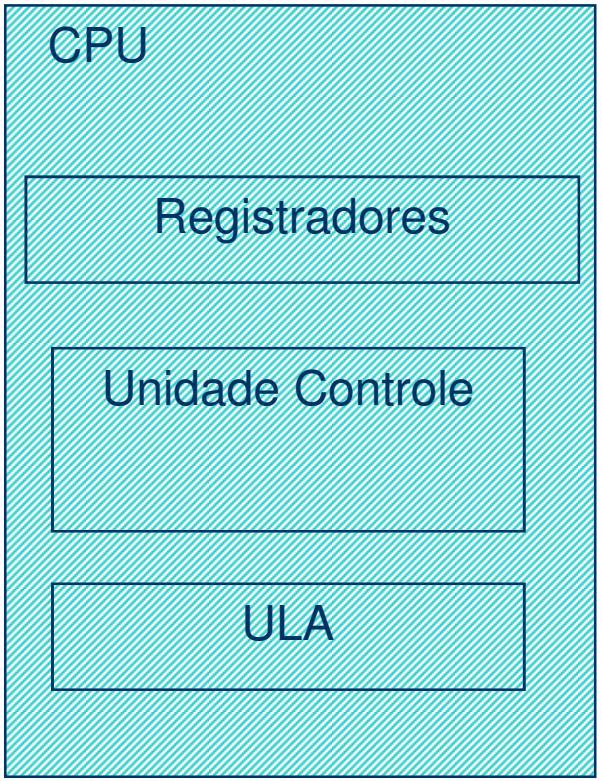


Porta NAND


Contadores
Registradores
Flip-flops
Microprocessador
Somadores
Codificadores
Decodificadores

Microprocessador

É a **CPU** de um computador construído num único Circuito Integrado. Contém essencialmente a **unidade de controle**, a **unidade lógica e aritmética** e **registradores (Acumulador e outros)**. Precisa de periféricos tais como memória e unidade de entrada e saída, para a formação de um sistema mínimo.



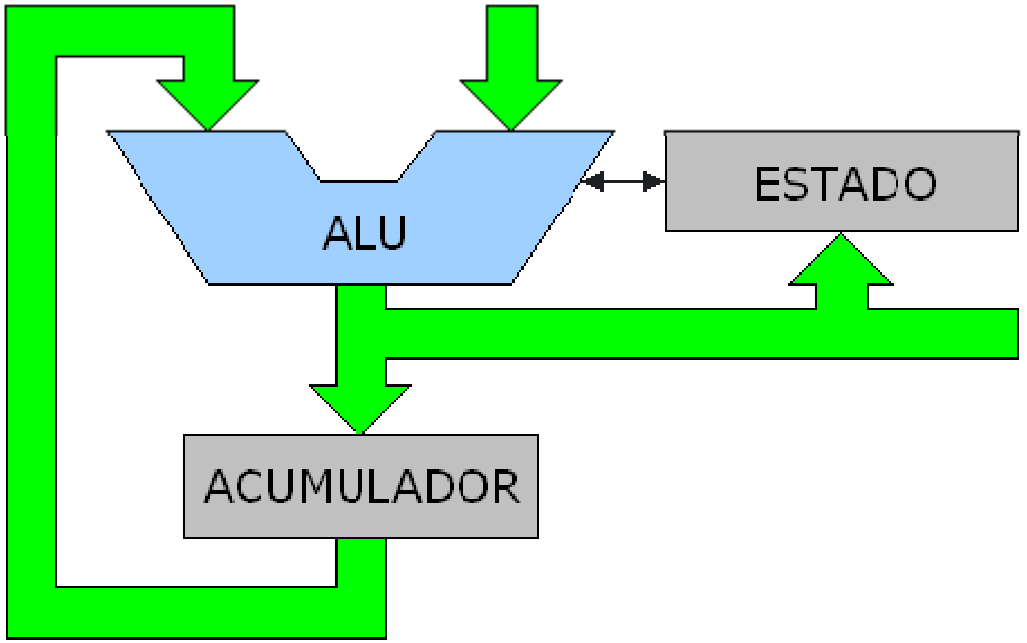
Unidade Central de Processamento (CPU)

Unidade de Controle (UC) - tem por função básica o controle das demais unidades da CPU através de sinais para transferência de dados e sinais de sincronismo entre unidades.

Por exemplo: a unidade de controle é responsável por enviar um sinal de leitura de uma instrução da memória ROM e sinais para transferência dessa instrução para a unidade de decodificação e, na sequência, sinais para as outras unidades que executarão a instrução, em uma sequência lógica.

Unidade Central de Processamento (CPU)

Unidade Lógica e Aritmética (ULA ou ALU) - realiza funções básicas de processamento de dados (adição, subtração, funções lógicas, etc.).



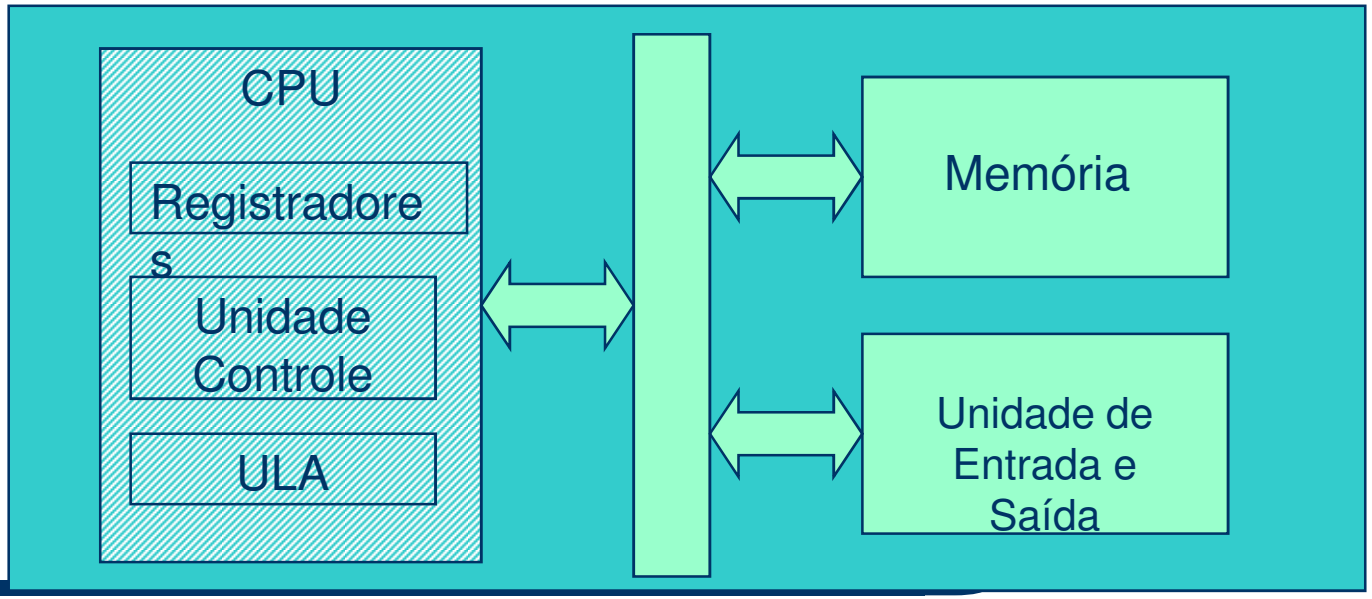
Unidade Central de Processamento (CPU)

Registradores - São usados para o armazenamentos internos da CPU. Existem diversos registradores na CPU e o principal deles é chamado de **Acumulador**.

Os registradores são construídos com flip-flops, que podem reter (armazenar) dados. O acumulador contém um dos dados usados na operação que se deseja e ainda o resultado da operação, que substitui o dado original.

Microcontrolador

Computador completo construído num único Circuito Integrado.
Os microcontroladores são normalmente utilizados para aplicações específicas, tais como sistemas de segurança, controle de velocidade de um motor e outros. Eles contêm, dentre outras unidades, portas de entrada e saída seriais e paralelas, temporizadores, controles de interrupção, memórias RAM e ROM.



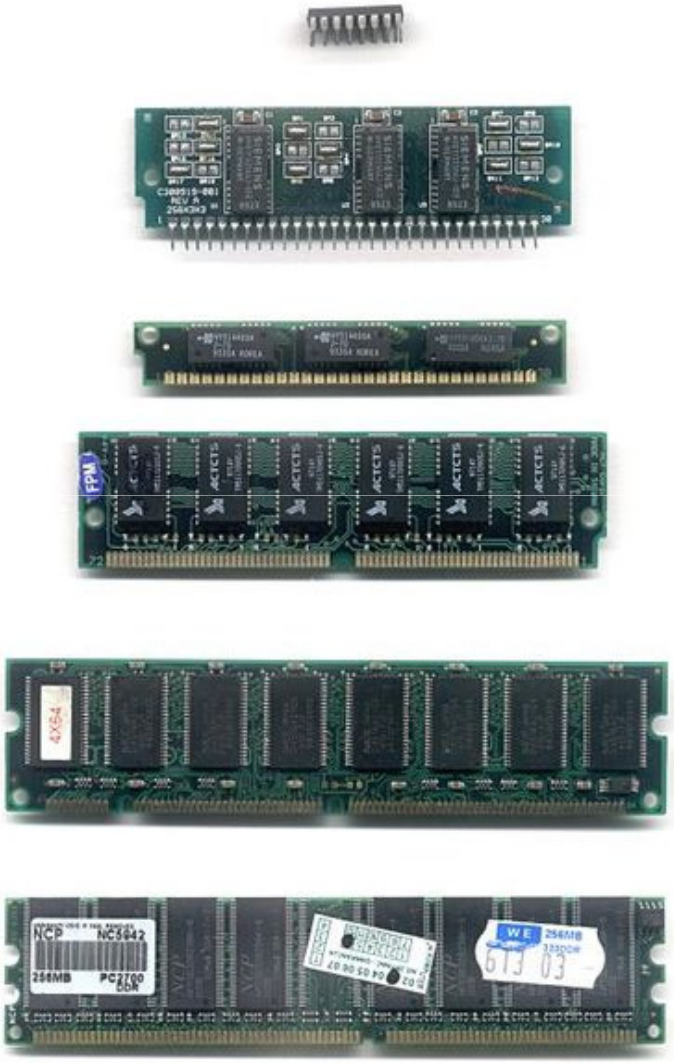
Memória **RAM – Permite a leitura e a gravação de dados.**

Memória Dinâmica (**DRAM**) – Baixa densidade, mas lenta. Capacitores com circuitos com "atualização de dados - *refresh*".

Memória estática (**SRAM**) – Alta densidade. Rápida. Baseada em Flip-flops.

Memória **CACHE** - Pequena quantidade de memória RAM estática (SRAM) usada para acelerar o acesso à memória principal (RAM dinâmica).

Quando há necessidade de transferir dados da (para) memória dinâmica, estes são antes transferidos para a memória cache



Memória ROM (*Read-Only Memory*)

Memória que permite apenas a leitura, ou seja, as suas informações são gravadas pelo fabricante uma única vez (no caso do tipo PROM) e após isso não podem ser alteradas ou apagadas, somente acessadas.

Alguns tipos de memória ROM:

PROM (*Programmable Read-Only Memory*) – Podem ser escritas com dispositivos especiais mas não podem mais ser apagadas.

EPROM (*Erasable Programmable Read-Only Memory*) – Podem ser apagadas pelo uso de radiação ultravioleta permitindo sua reutilização.

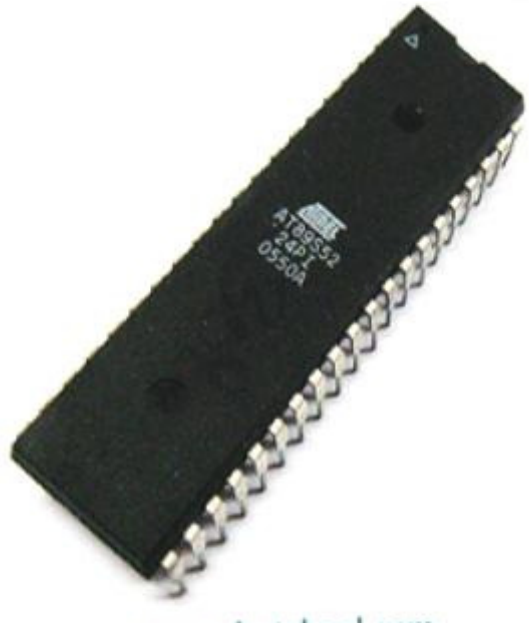
Exemplo para o caso do 8051:
Microcontrolador **8751**

EEPROM (***E**lectrically **E**rasable **P**rogrammable **R**ead-**O**nly Memory*) – O seu contéudo pode ser modificado eletricamente.

Exemplo: EEPROM AT28C64 – Memória com 8 K de 8 bits

Memória **FLASH** – São semelhantes às EEPROMs, porém mais rápidas e de menor custo.

Exemplo: Microcontrolador AT89S52



Unidade de Entrada e Saída (I/O)

A entrada de dados de um microprocessador (via teclado, mouse ou outros dispositivos) e a saída de dados (via vídeo, impressora ou outros) exige circuito integrado adicional como interface.

Dois exemplos:

CI 8156 - RAM e porta de entrada e saída e

CI 8355 – ROM e portas de entrada e saída)

O microcontrolador já possui essas unidades internamente.

Definições Básicas de Microprocessadores

Índice de Desempenho de Processadores

O aumento de desempenho (velocidade de processamento) de processadores está relacionado com os seguintes aspectos:

- Aumento de clock
- Aumento do número interno de bits
- Aumento do número externo de bits
- Redução do número de ciclos para executar cada instrução
- Aumento da capacidade e velocidade da memória cache
- Execução de instruções em paralelo

Aumento de Clock

O sinal de clock é responsável pelo sincronismo entre as unidades de processamento internas ao microprocessador e pelas unidades externas. Quanto maior a frequência de clock mais rápido o processamento. No entanto, não se pode aumentar de forma indefinida essa frequência. Isso pode causar falhas de processamento e sobreaquecimento. O aumento depende de pesquisas com o objetivo de reduzir o tamanho dos componentes básicos do microprocessador e aumento da quantidade de componentes, sem perda de estabilidade no funcionamento.

Aumento do número interno de bits

Uma maior quantidade de bits dos registradores e dos barramentos internos permite a movimentação de uma maior quantidade de dados por unidade de tempo, aumentando o desempenho do microprocessador.

Aumento do número externo de bits

Um número maior de bits externos permite a movimentação de uma maior quantidade de dados por unidade de tempo com os periféricos, tais como memória, unidade de entrada e saída, controlador de acesso direto à memória (DMA).

Redução do número de ciclos para executar cada instrução

A execução de uma instrução normalmente é feita em duas etapas: **busca** (onde a instrução é transferida da memória para a unidade de decodificação) e **execução** (onde os sinais de controle ativam, em uma sequência lógica, todas as unidades envolvidas na execução).

No microprocessador 8085 as instruções mais rápidas são executadas em quatro ciclos de clock; as mais lentas, em até 16 ciclos de clock. A redução do número de ciclos de clock na execução de uma instrução torna o processamento mais rápido.

Aumento da capacidade e velocidade da memória cache

Como já foi dito anteriormente, ao longo dos anos, o aumento de velocidade de processamento dos microprocessadores tem sido muito maior do que o aumento da velocidade de acesso à memória principal. Assim, a velocidade de acesso à memória principal torna-se um limitador de desempenho dos processadores. Em razão desse problema foi criada a memória cache. A memória cache (constituída de memória RAM estática) é usada para acelerar a transferência de dados entre a CPU e a memória principal (constituída de RAM dinâmica, de menor volume, porém mais lenta). O aumento da capacidade e da velocidade da memória cache resulta no aumento da velocidade de transferência de dados entre a CPU e a memória principal e, consequentemente, resulta no aumento do desempenho global do sistema.

Execução de instruções em paralelo

O microprocessador 8085 compartilha um barramento comum entre suas unidades internas e seus periféricos, o que significa dizer que não permite a execução simultânea de duas operações que utilizem o barramento. Assim, apenas uma instrução é executa por vez. Uma arquitetura que permita que duas ou mais operações sejam executadas simultaneamente torna o processamento mais rápido.

MIPS - *Millions of Instructions Per Seconds* (Milhões de Instruções Por Segundo): É uma unidade de desempenho do microprocessador.

FLOPS - *FLOating point instructions Per Seconds* (Instruções com Ponto Flutuante Por Segundo). É também uma unidade de desempenho do microprocessador. Indica a capacidade de trabalhar com números decimais.

Representação em Ponto Fixo - Sistema numérico no qual o ponto está implicitamente fixo (à direita do digito mais a direita).

Representação em Ponto Flutuante - Sistema numérico no qual um número real é representado por um par distinto de numerais: uma mantissa (ou significante) e um expoente. Possibilita representação de números fracionários.

Bit

Abreviatura para “Binary Digit”, ou, Dígito Binário. Pode assumir valor 0, que corresponde a tensão O V, ou 1, que representa normalmente uma tensão de 5 V ou 3,3 V.

Byte

Conjunto de 8 bits. É a **unidade básica de dados nos computadores**, que também utilizam alguns múltiplos de 8, tais como 16 bits (Word) e 32 bits (Dword).

Set de Instruções

Conjunto de Instruções - Conjunto de Mnemônicos (siglas que fazem lembrar uma ação) que representam todas as instruções do processador. Cada processador possui o seu set de instruções particular. O microprocessador 8085 possui 74 instruções.

BIOS

Basic **I**nput/**O**utput **S**ystem – É o conjunto mínimo de instruções necessárias para a inicialização do computador. Também gerencia o fluxo de dados entre o sistema operacional do computador e os dispositivos periféricos conectados.

Alguns Exemplos de Aplicação de Microprocessadores e Microcontroladores

Microcomputadores, Calculadoras, Relógios Digitais,

Controle de Fornos Micro-ondas, Lavadora de Roupas,

Video Games e outros brinquedos, Controle de Motores,

Controle de Tráfego, Alarmes e Sistemas de Segurança,

Telefone Celular.

Alguns Exemplos de Projetos Finais Implementados Usando Microcontroladores na EEEC

Controle Escalar de Motor de Indução, Controle de Motor de Corrente Contínua, Sistema de Controle de Portão Eletrônico, Sistema de Controle de Acesso ao Laboratório, Sistema de Controle de Presença em Sala de Aula (Diário Eletrônico), Sistema de Controle de Umidade e Temperatura de uma Sala, Sistema de Monitoramento de Batimentos Cardíacos, Sistema de Monitoramento de Temperatura e Umidade de Estufas, Sistema de Controle de Acesso a Estacionamentos, Sistema de Monitoramento de Tarifação de Telefone, Sistema de Controle para Centrífugas de Apiários, Sistema de Suporte para Deficientes Visuais.

Sistemas de Numeração

Sistema Posicional × Sistema não Posicional

$$Valor = \sum_{i=0}^{n-1} d_i B^i$$

Exemplos de representação: 1011_2 , 1011_{10} , 1011_{16}, 1011_8

Decimal: $$Valor = \sum_{i=0}^{3} d_i 10^i = 1\times10^0 + 1\times10^1 + 0\times10^2 + 1\times10^3$$
$$= 1 + 10 + 0 + 1000 = 1011$$

Binário: $$Valor = \sum_{i=0}^{3} d_i 2^i = 1\times2^0 + 1\times2^1 + 0\times2^2 + 1\times2^3$$
$$= 1 + 2 + 0 + 8 = 11$$

Sistema Posicional × Sistema não Posicional

$$Valor = \sum_{i=0}^{n-1} d_i B^i$$

Exemplos de representação: 1011_2 , 1011_{10} , 1011_{16}, 1011_8

Hexadecimal:
$$Valor = \sum_{i=0}^{3} d_i 16^i = 1\times16^0 + 1\times16^1 + 0\times16^2 + 1\times16^3$$
$$= 1 + 16 + 0 + 4096 = 4113$$

Octal:
$$Valor = \sum_{i=0}^{3} d_i 8^i = 1\times8^0 + 1\times8^1 + 0\times8^2 + 1\times8^3$$
$$= 1 + 8 + 0 + 512 = 521$$

Sistema Binário - O sistema binário é o sistema de numeração que o computador entende.

Utiliza 2 dígitos: 0 e 1 ou (OFF e ON) ou (0V e 5V) ou (0V e 3,3V).

Exemplo: $1\ 1\ 0\ 0\ 1\ 0\ 1\ 1_2$

1º dígito: Armazena o equivalente a 2^0 (1). No ex.: 1×2^0

2º dígito: Armazena o equivalente a 2^1 (2). No ex.: 1×2^1

3º dígito: Armazena o equivalente a 2^2 (4). No ex.: 0×2^2

…

8º dígito: Armazena o equivalente a 2^7 (128): No ex.: 1×2^7

A soma destas parcelas resulta no seguinte equivalente decimal:

$1 + 2 + 0 + 8 + 0 + 0 + 64 + 128 = 203_{10}$

No sistema binário a ponderação é dada pelo número 2 elevado à potência representada pela coluna, sendo que a 1ª coluna é 0, a segunda coluna é 1 e assim sucessivamente.

1	1	0	0	1	0	1	1

1 kbyte = 2^{10} = 1.024 bytes

1 Mbyte = 2^{10} x 2^{10} = 1.048.576 bytes = 1.024 kbytes;

1 Gbyte = 2^{10} x 2^{10} x 2^{10} = 1.073.741.824 bytes = 1.024 Mbytes

Sistema BCD (Binary-Coded Decimal) – O Sistema BCD é o sistema em que se combina o sistema binário e o sistema decimal.

É utilizado como formato de saída de instrumentos.

Utiliza 2 dígitos: 0 e 1 que são dispostos em grupos de 4 dígitos, utilizados para representar um dígito decimal (número 0 até 9).

A representação de um número maior que 9 deve ser feita por outro grupo de 4 bits, com a ponderação dada pelo sistema decimal.

Exemplo: 973_{10} = 1001 0111 0011.

Note a diferença entre este valor e o valor do número binário

$1001\ 0111\ 0011_2 = 2419_{10}$

Sistema Octal - O Sistema Octal é baseado nos mesmos princípios do decimal e do binário, apenas utilizando base 8.

Utiliza 8 dígitos: 0 a 7.

Exemplo: 3207_8

1º dígito: Armazena o equivalente a 8^0 (1). No ex.: 7×8^0
2º dígito: Armazena o equivalente a 8^1 (8). No ex.: 0×8^1
3º dígito: Armazena o equivalente a 8^2 (64). No ex.: 2×8^2
4º dígito: Armazena o equivalente a 8^3 (512). No ex.: 3×8^3

O equivalente decimal é: $7 + 0 + 128 + 1536 = 1671_{10}$

Sistema Hexadecimal - O Sistema Hexadecimal é baseado nos mesmos princípios do decimal e do binário, apenas utilizando base 16.

Utiliza 16 dígitos: 0 a 9, A, B, C, D, E, F.

Exemplo: 20DH ou 20Dh ou $20D_{16}$

1º dígito: Armazena o equivalente a 16^0 (1). No ex.:13×16^0

2º dígito: Armazena o equivalente a 16^1 (16). No ex.: 0×16^1

3º dígito: Armazena o equivalente a 16^2 (256). No ex.: 2×16^2

O equivalente decimal é: $13 + 0 + 512 = 525_{10}$

Conversão de Base

O sistema hexadecimal é mais fácil de trabalhar que o sistema binário e é geralmente utilizado para escrever endereços.

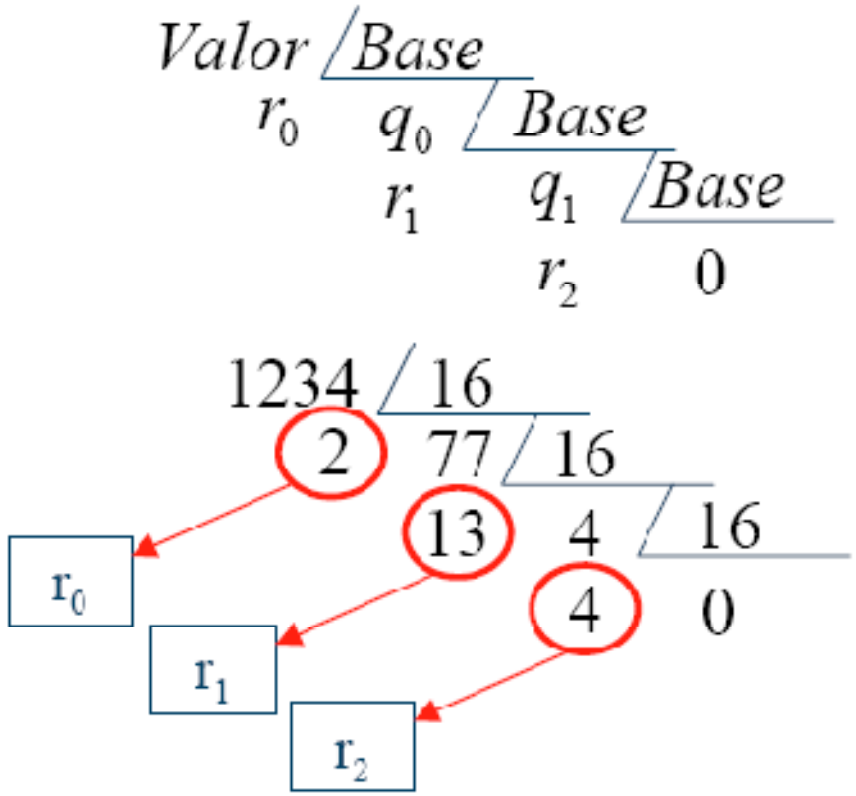
Na conversão de hexadecimal para binário, cada dígito hexadecimal é convertido em 4 dígitos binários equivalentes.

Exemplo: 7 D 3 F_{16} = $0111\ 1101\ 0011\ 1111_2$

Na conversão de binário para hexadecimal, cada grupo de 4 dígitos binários é convertido em 1 dígito hexadecimal equivalente.

Ex.: $\mathbf{1010000110111000_2}$ **=** $1010\ 0001\ 1011\ 1000_2 = A\ 1\ B\ 8_{16}$

Conversão de Base



Representação: $r_2\ r_1\ r_0$

Valor hexadecimal correspondente a 1234: 4D2h

Conversão de Base (algoritmo genérico)

$$\begin{array}{r|l} Valor & Base \\ r_0 & q_0 \end{array} \qquad \begin{array}{r|l} q_0 & Base \\ r_1 & q_1 \end{array} \qquad \begin{array}{r|l} q_1 & Base \\ r_2 & 0 \end{array}$$

A divisão termina quando o quociente é zero

Se $q_0 = 0$ → Valor = $q_0 B + r_0 = 0.B + r_0 = r_0$

Se $q1 = 0$ → Valor = $q_0 B + r_0$

$q_0 = q_1 .B + r_1 = 0.B + r_1 = r_1$

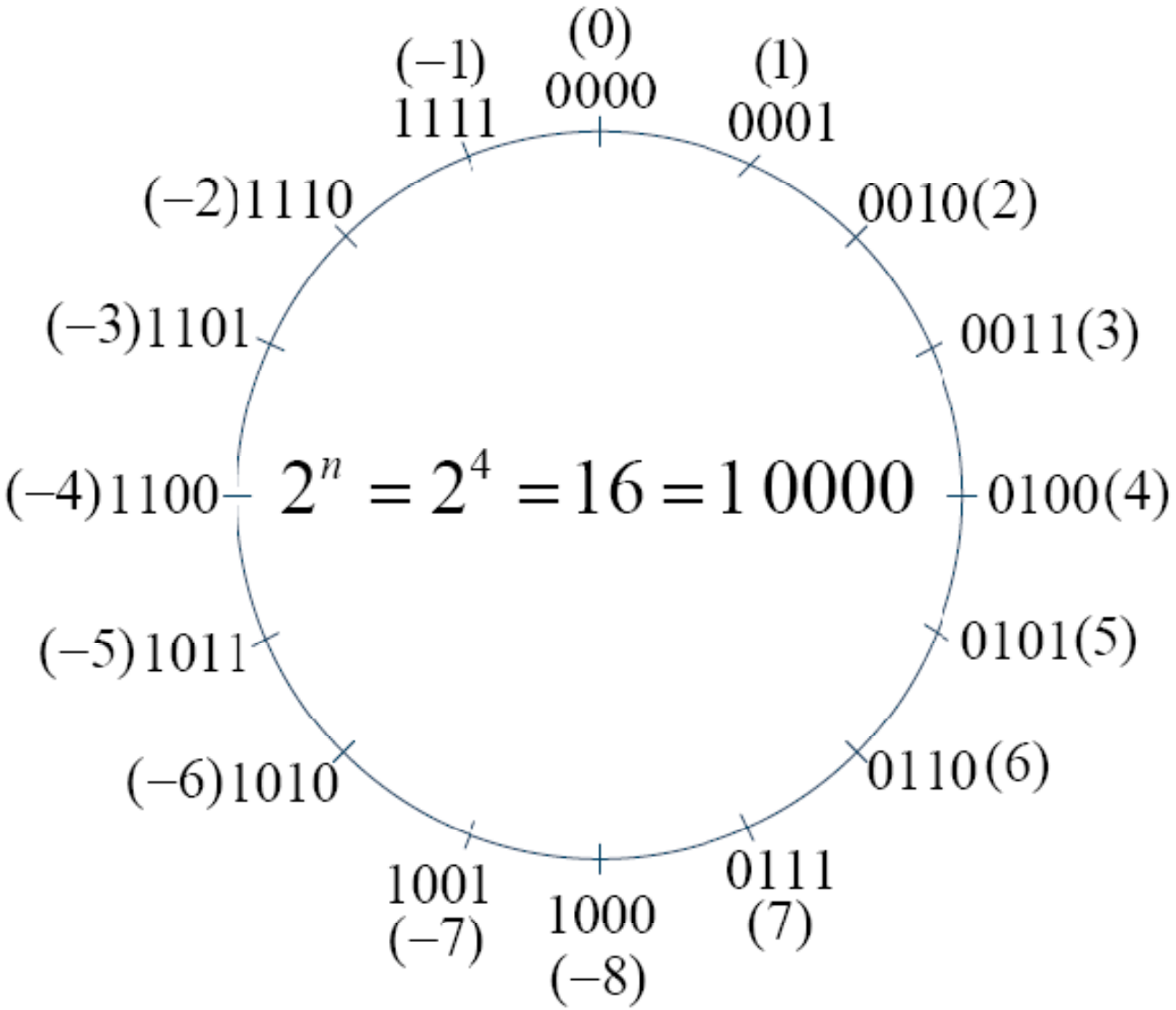
ou, Valor = $r_1 .B + r_0$ → Representação: $r_1\ r_0$

Se $q2 = 0$ → Valor = $q_0 B + r_0$, $q_0 = q_1 .B + r_1$

$q_1 = q_2 .B + r_2 = 0.B + r_2 = r_2$

Valor = $r_2 B^2 + r_1 B + r_0$ → Representação: $r_2\ r_1\ r_0$

Representação de números positivos e negativos



Valor Simétrico de um Número

Número binário:

$-a$ = (complemento de 1 de a) + 1 =

= complemento de 2 de a = $2^n - a$

Número decimal:

$-a$ = (complemento de 9 de a) + 1 =

= complemento de 10 de a = $10^n - a$

Subtração Usando Adição

Número binário:

$a - b = a +$ (complemento de 2 de b) →

$a - b = a + (2^n - b)$

Exemplo para um número binário de 4 dígitos:

$a - 1 = a + (2^4 - 1) = a + (10000 - 0001) = a + 1111$
$a - 3 = a + (2^4 - 3) = a + (10000 - 0011) = a + 1101$

Se $a = 1001\ (9_{10})$ → $a - 1 = 1001 - 0001 = 1000$
→ $a - 3 = 1001 - 0011 = 0110$

ou

$a - 1 = 1001 + 1111 = 1\ \underline{1000}$ (despreza-se o quinto dígito)
$a - 3 = 1001 + 1010 = 1\ \underline{0110}$ (número é de 4 dígitos)

Subtração Usando Adição

Número decimal:

$a - b = a +$ (complemento de 10 de a)

$a - b = a + (10^n - b)$

Exemplo para um número decimal de 2 dígitos:

$a - 1 = a + (10^2 - 1) = a + (100 - 1) = a + 99$
$a - 3 = a + (10^2 - 3) = a + (100 - 3) = a + 97$

Se $a = 94 \rightarrow a - 1 = 94 - 1 = 93$
$\rightarrow a - 3 = 94 - 3 = 91$

ou

$a - 1 = 94 + 99 = 1\ \underline{93}$ (despreza-se o terceiro dígito)
$a - 3 = 94 + 97 = 1\ \underline{91}$ (o número é de 2 dígitos)

Operações Aritméticas no Microprocessador

Adição → Adição direta

Subtração → Adição com o complemento de 2

Multiplicação → Várias adições

Divisão → Várias adições com complemento de 2

Arquitetura RISC x Arquitetura CISC

Objetivo básico de qualquer arquitetura de microprocessador:

aumentar o desempenho do processador reduzindo o tempo de execução de tarefas.

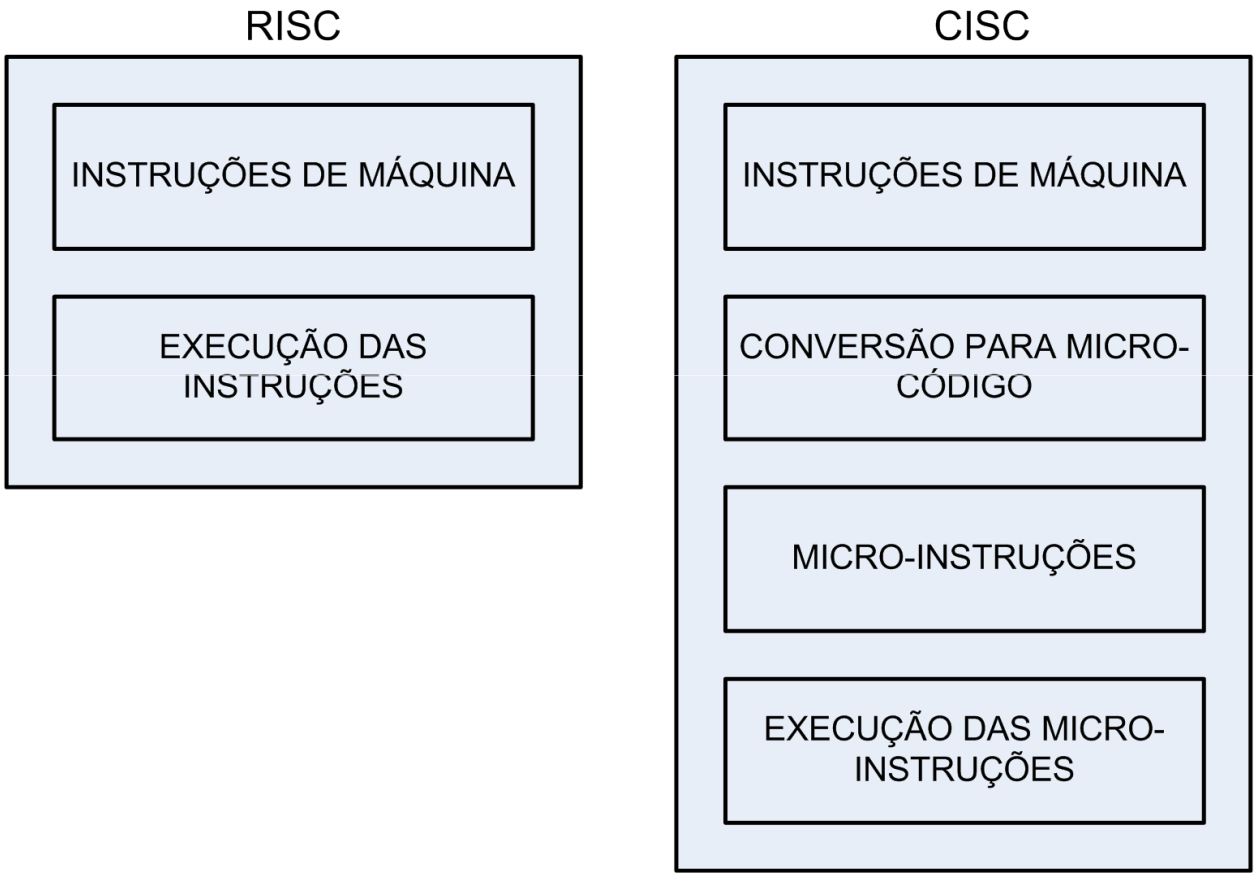
Arquiteturas em uso nos computadores atuais:

- **CISC** – Complex Instruction Set Computing (Computador com Conjunto Complexo de Instruções) Exemplos: Intel e AMD
- **RISC** – Reduced Instruction Set Computing (Computador com Conjunto Reduzido de Instruções) Exemplos: PowerPC (da Apple, Motorola e IBM), SPARC (SUN) e MIPS R2000
- **Híbrida** – Combinação de ambas arquiteturas. Exemplo: Pentium Pro. O núcleo mais interno usa filosofia RISC.

Algumas Características RISC X CISC:

RISC	CISC
Instruções básicas executadas em apenas 1ciclo de via de dados	No mínimo 4 ciclos de clock para executar uma instrução
Uso reduzido da memória	Muitas instruções com acesso à memória
Uso reduzido das vias de dados	Uso intenso das vias de dados
Uso preferencial de registradores, ao invés de memória – muitos registradores	Uso de um número menor de registradores que a arquitetura RISC
Não há necessidade de decodificação das instruções antes de executá-las	Ciclo de busca inclui busca na memória e identificação em decodificadores
Instrução semelhantes às micro-instruções da arquitetura CISC. As instruções são executadas diretamente no hardware	Uso de micro-instruções gravadas no processador. Necessidade de interpretação das instruções
Instruções simples e em número limitado	Instruções complexas. Várias instruções
Programa compilado tem maior número de instruções em assembly. Uso maior de memória	Menor número de instruções assembly, porém mais lento na execução

Etapas de execução de uma instrução:



CONJUNTO DE INSTRUÇÕES:

Grupos de instruções mais comuns em processadores de qualquer arquitetura:

Instruções de desvio (No CISC o valor de retorno é guardado na pilha; no RISC é guardado em um registrador.

Instruções de transferência entre registradores e memória (No RISC: load/store; no CISC: load, store, mov etc)

Instruções de transferência entre registradores

Instruções de transferência entre posições de memória

Operações aritméticas (soma, subtração ...)

Operações lógicas (and, or, not, rotação ...)

CICLO DE EXECUÇÃO:

RISC → As instruções são executadas em um único ciclo de via de dados. São instruções muito parecidas com as micro-instruções da arquitetura CISC. Não precisam de decodificação.

Não é possível ter instruções de multiplicação e divisão, por exemplo, por exigir muitos ciclos para execução. Multiplicações são resolvidas com adições e deslocamentos.

CISC → Antes de executar uma instrução, há necessidade de busca da instrução na memória e de decodificação. Utiliza-se micro-códigos gravados no processador, para a execução das instruções.

MEMÓRIA E REGISTRADORES:

RISC → Possui uma quantidade muito grande de registradores (em média 512 – com 32 visíveis por vez: 8 para variáveis globais e ponteiros, 8 para parâmetros de entrada, 8 para variáveis locais e 8 para parâmetros de saída).

Número reduzido de acesso à memória (o acesso à memória torna o processamento mais lento). Alocação de variáveis em registradores.

Um ou dois modos de endereçamento para acesso à memória

CISC → Possui um número reduzido de registradores, comparado com o RISC. Alocação de variáveis em posições de memória, ao invés de registradores.

Vários modos de endereçamento para acesso à memória.

MICRO-CÓDIGOS:

RISC → As instruções geradas por um compilador para uma máquina RISC são executadas diretamente no hardware, sem o uso de micro-códigos. A ausência de interpretação contribui para o aumento da velocidade de execução.

CISC → As instruções básicas são gravadas na forma de micro-códigos, que atuam no hardware estabelecendo os passos de cada instrução. Há necessidade de busca e decodificação das instruções.

O programa compilado tem uma quantidade menor de instruções assembly do que um programa RISC, mas é mais lento na execução.

PIPELINE:

Técnica usada para acelerar a execução de instruções. A cada ciclo de clock, enquanto uma instrução está na etapa de execução, a instrução seguinte está sendo buscada. O resultado global é que, a cada ciclo, uma nova instrução é iniciada e uma instrução é encerrada. No caso mostrado a instrução B faz referência à memória.

Ciclos	1	2	3	4	5
Busca da instrução	A	B	C	D	E
Execução da instrução		A	B	C	D
Referência à memória				B	

PIPELINE:

Enquanto a instrução A precisa de apenas um ciclo para busca e um para execução, a instrução B precisa de dois ciclos para execução. Caso a instrução B interfira na etapa de execução da instrução C (por exemplo, usando o mesmo registrador ou quando a instrução C precisa do resultado da instrução B) é necessário aguardar o término da instrução B antes de executar a instrução C.

Ciclos	1	2	3	4	5	6
Busca da instrução	A	B	C	NOP	D	E
Execução da instrução		A	B	NOP	C	D
Referência à memória				B		

VANTAGENS RISC:

- Velocidade de execução
- O uso de pipeline torna os processadores RISC duas a quatro vezes mais rápidos que um CISC de mesmo clock
- Simplicidade de Hardware
- Ocupa menos espaço no chip, devido ao fato de trabalhar com instruções simples.
- Instruções de máquina simples e pequenas, o que aumenta sua performance.

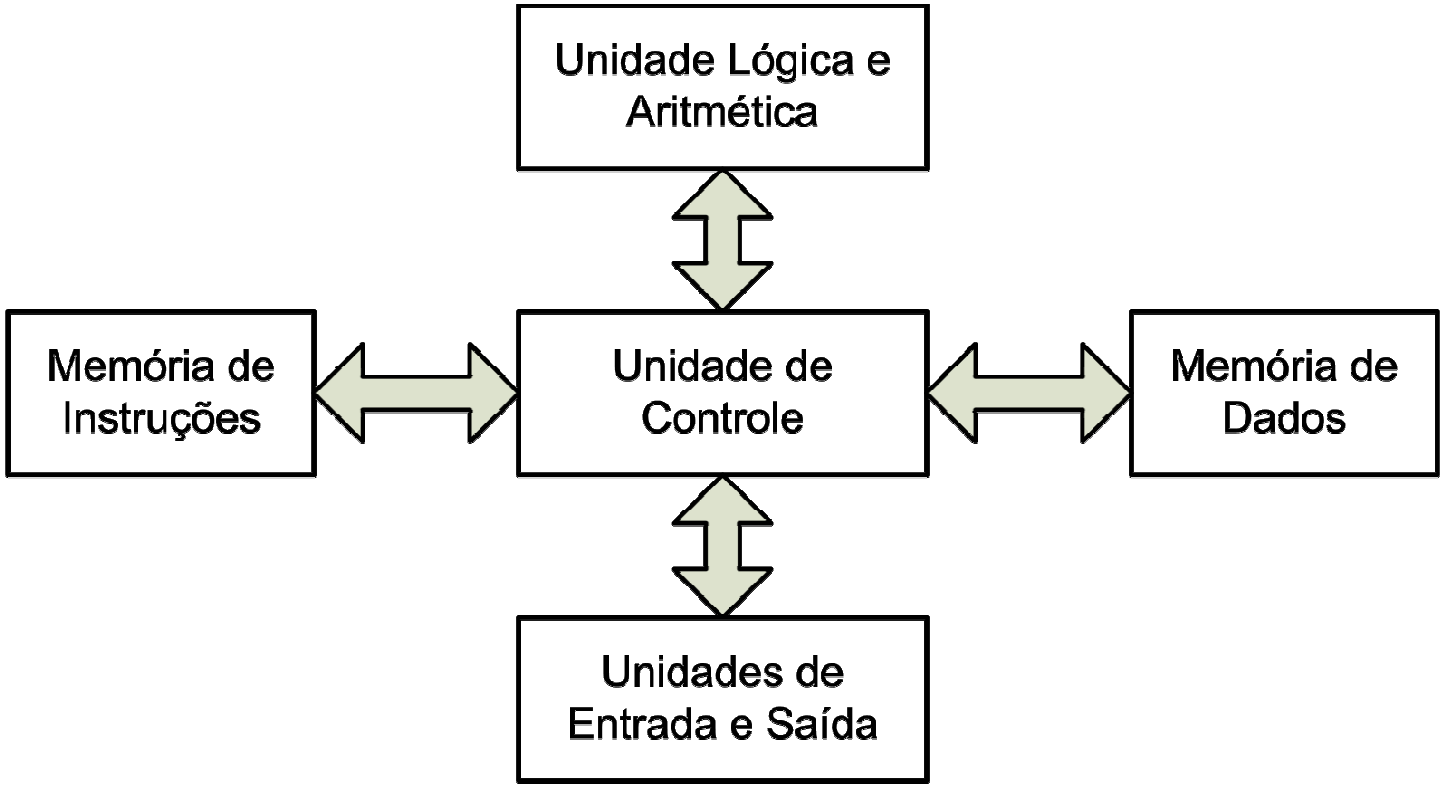
DESVANTAGENS RISC:

• O desempenho de um processador RISC depende diretamente do código gerado pelo programador. Um código mal desenvolvido pode resultar em tempo de execução muito grande.

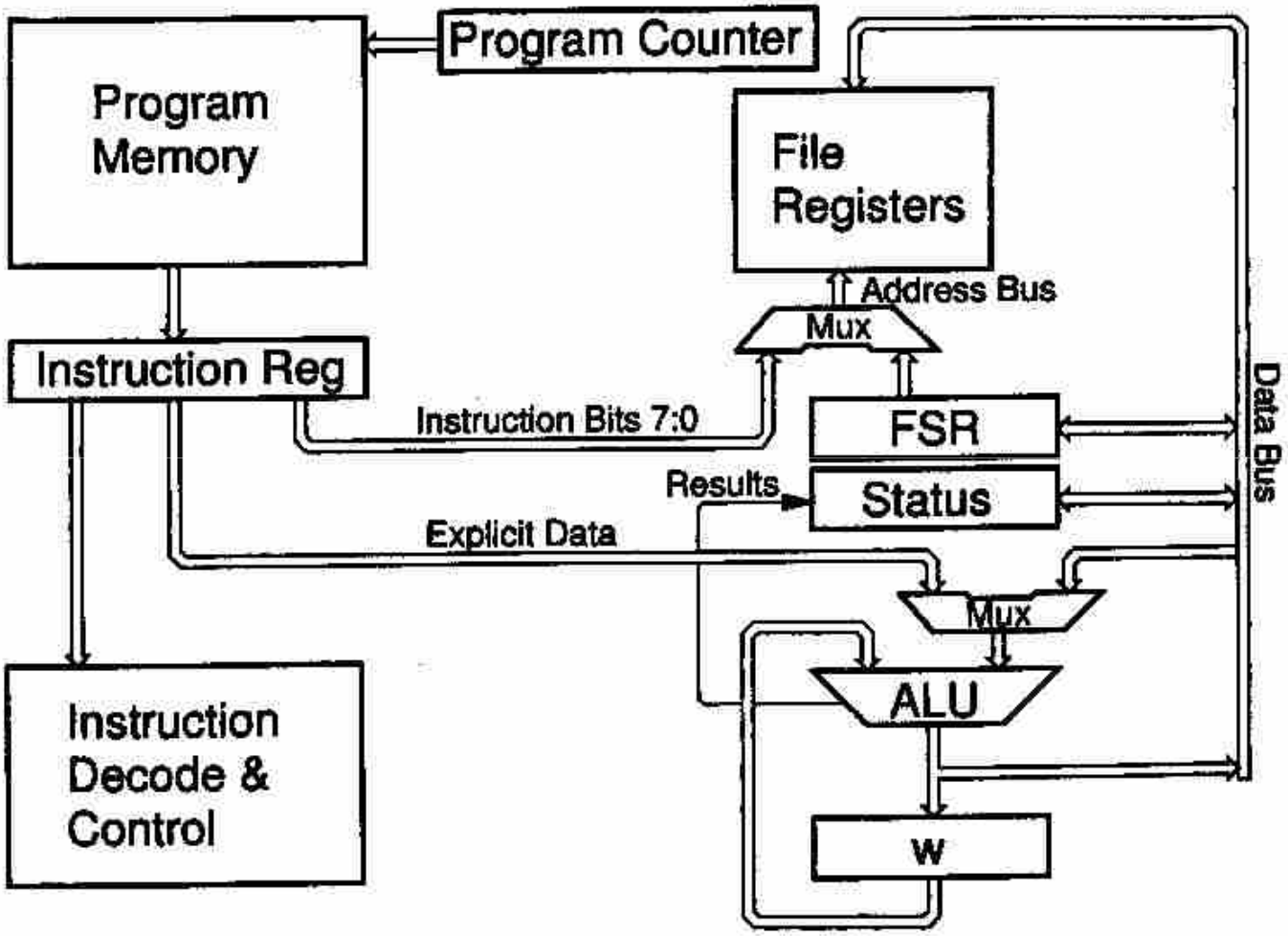
• Um programa originalmente compilado para uma máquina CISC tem um equivalente compilado para máquina RISC com uma quantidade muito maior de códigos assembly, ocupando um espaço maior na memória.

• A arquitetura RISC requer sistema de memória rápida para alimentar suas instruções. Normalmente possuem grande quantidade de memória cache interna, o que encarece o projeto.

Arquitetura RISC

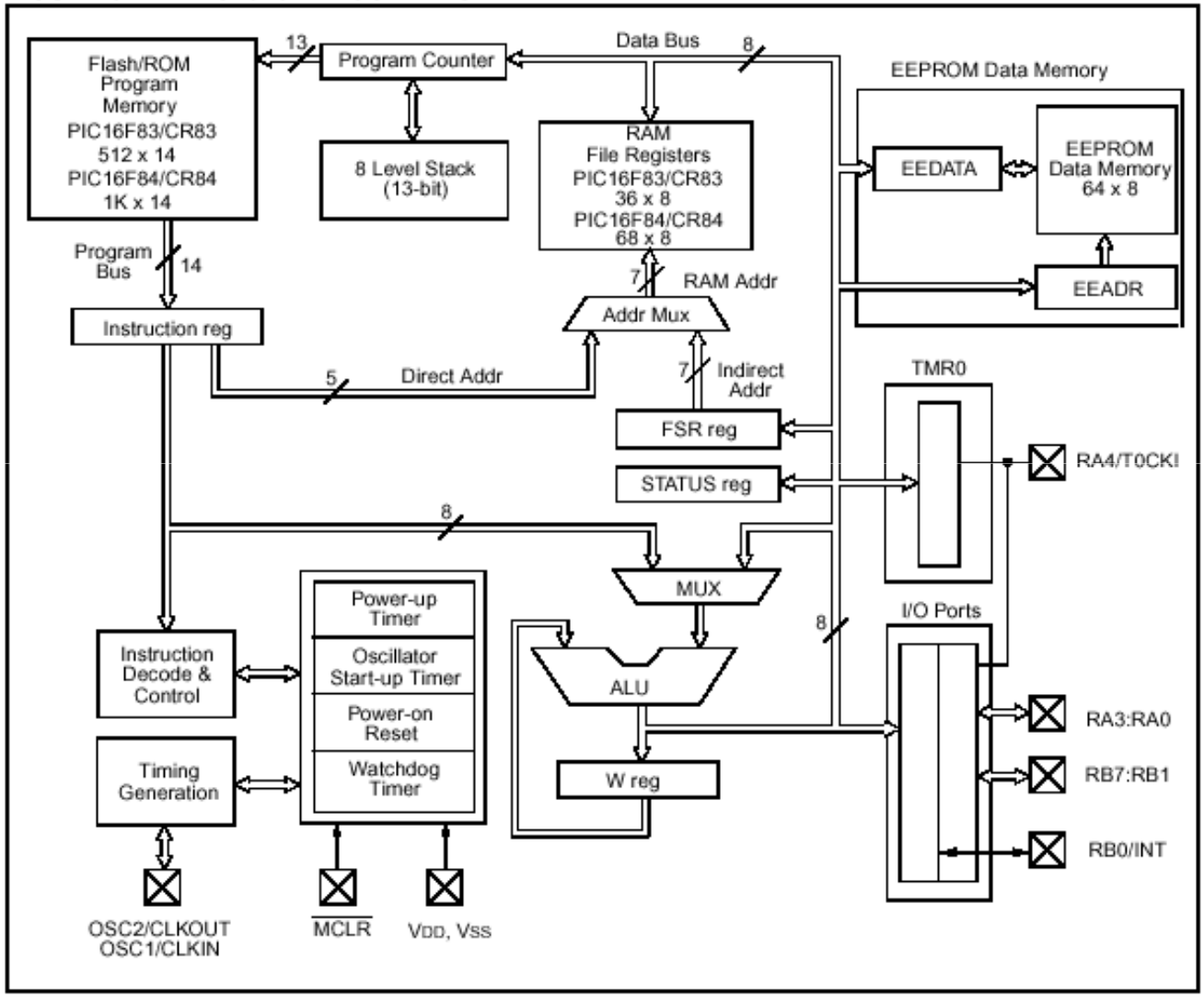


EXEMPLO DE ARQUITETURA RISC: MICROCONTROLADOR PIC



PICmicro® MCU Processor with Indexed addressing

FIGURE 3-1: PIC16F8X BLOCK DIAGRAM



Arquitetura, Características e Princípio de Funcionamento dos Microprocessadores 8085 e 8086/88

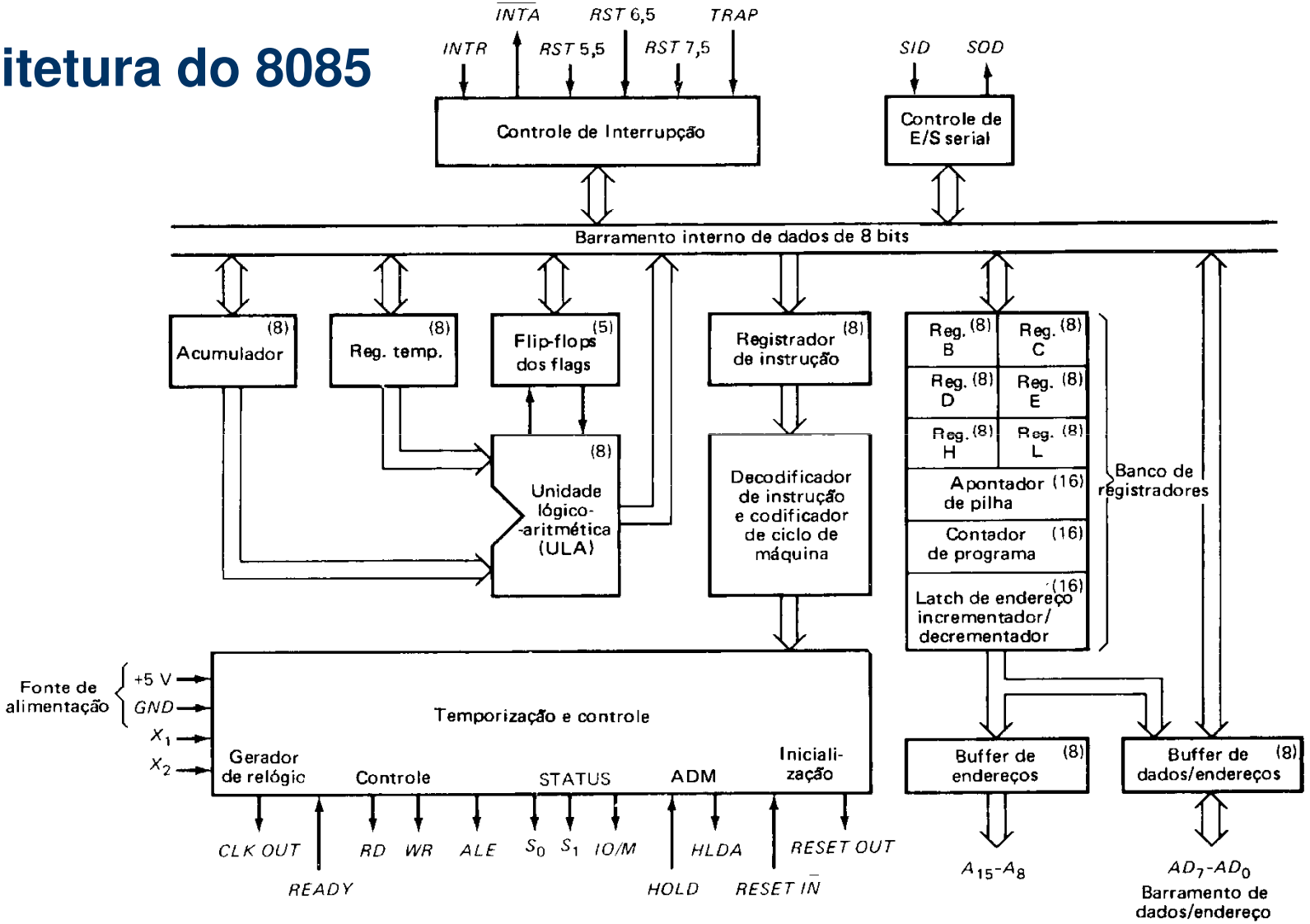
8085 × 8086 / 8088

Característica	Microprocessador 8085	Microprocessador 8088	Microprocessador 8086
Barramento de endereço	16 bits	20 bits	20 bits
Capacidade de endereçamento de memória	65.536 (64 kB)	1.048.576 (1 MB)	1.048.576 (1 MB)
Barramento de dados	8 bits	Interno: 16 bits Externo: 8 bits	Interno: 16 bits Externo: 16 bits
Manipulação de STRINGS	NÃO	SIM	SIM
Registradores Internos	8 bits e 16 bits	16 bits	16 bits
Uso de segmentação para endereçamento	NÃO	SIM	SIM
Aritmética Decimal completa	NÃO	SIM	SIM
Etapas de Busca e Execução	Em sequência: **Busca** → **Executa**	Unidades Independentes: Unidade de Interfaceamento com Barramento (**BIU**) – responsável pela **Busca** e Unidade de Execução (**EU**)	Unidades Independentes: Unidade de Interfaceamento com Barramento (**BIU**) – responsável pela **Busca** e Unidade de Execução (**EU**)

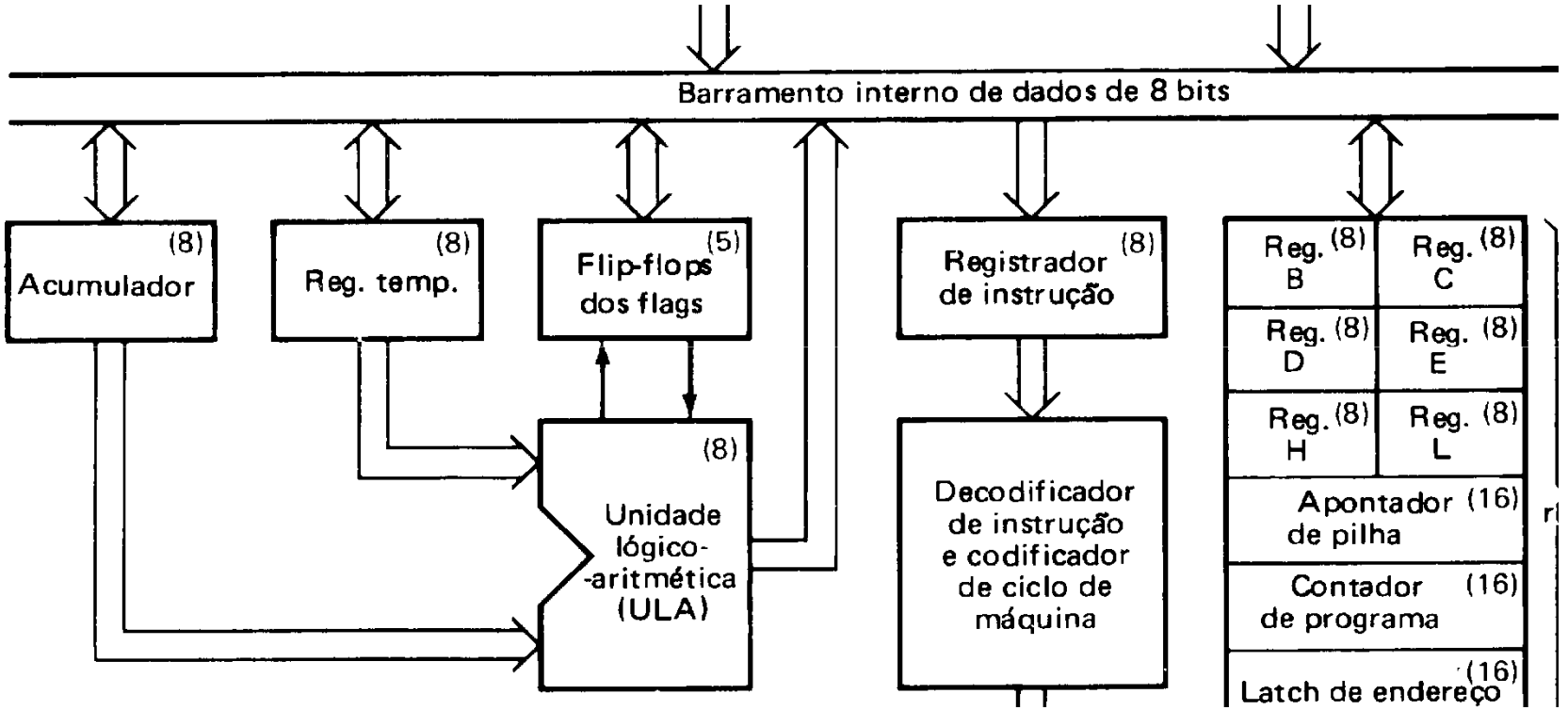
8085 × 8086 / 8088

Registradores do 8085				Registradores do 8088 / 8086			
	A	Acumulador		AH	AL (A)	AX – Acumulador Primário	
H	L	Apontador de dados		BH	BL	BX – Acumulador e Registrador Base	
B	C			CH	CL	CX – Acumulador e Contador	
D	E			DH	DL	DX – Acumulador e Endereçador de I/O	
SP		Apontador de pilha		SP		Apontador de pilha	
				BP		Apontador base – usado na pilha	
				SI		Índice da Fonte – usado para indexação	
				DI		Índice de Destino – usado para indexação	
PC		Contador de Programa		IP		Ponteiro de Instrução	
				CS	Segmento de Código	Registradores de segmento. São usados para a formação do endereço absoluto.	
				DS	Segmento de Dados		
				SS	Segmento de Pilha		
				ES	Segmento Extra		
	FLAGS	Registrador de Flags		FLAGS	Registrador de Flags		

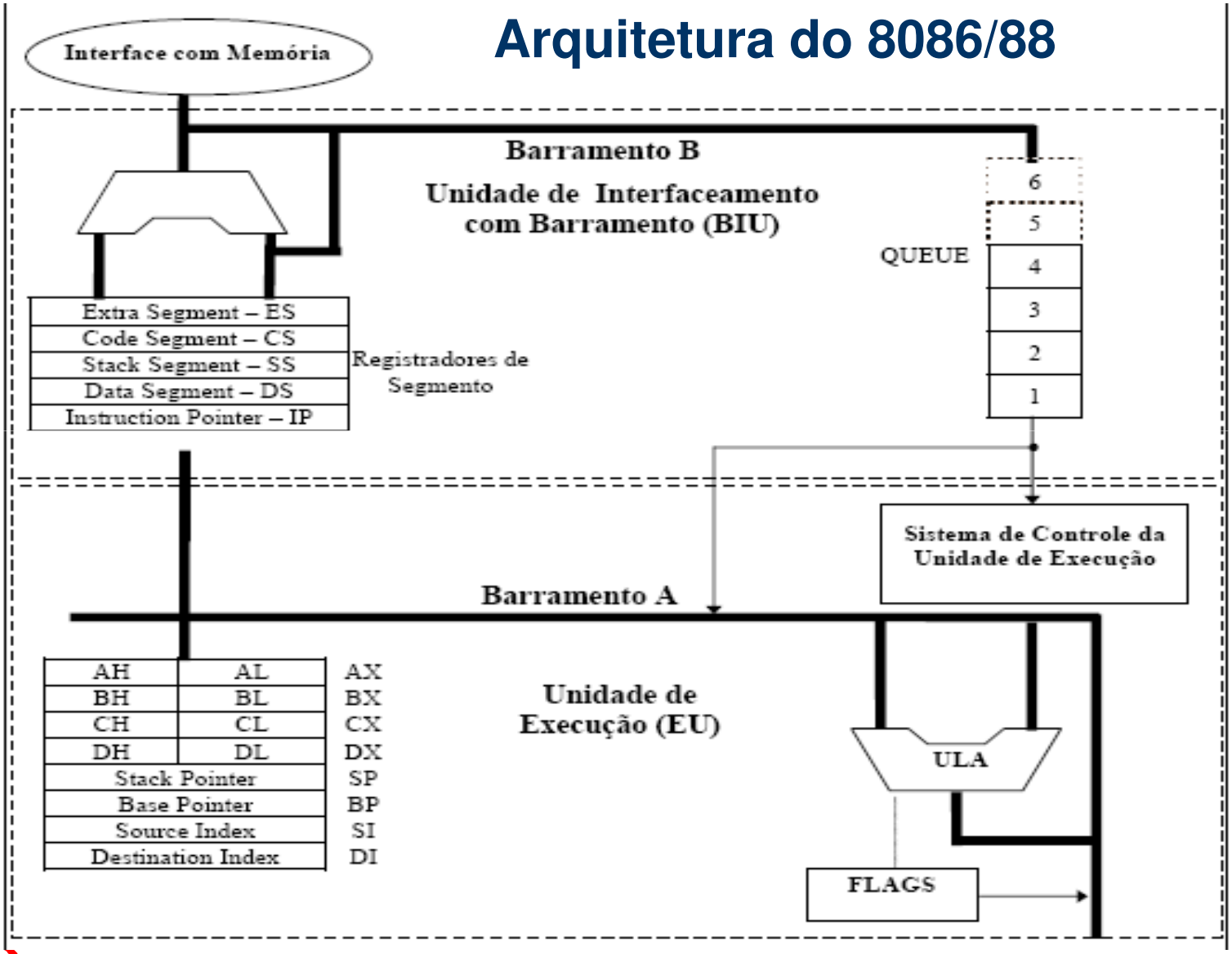
Arquitetura do 8085



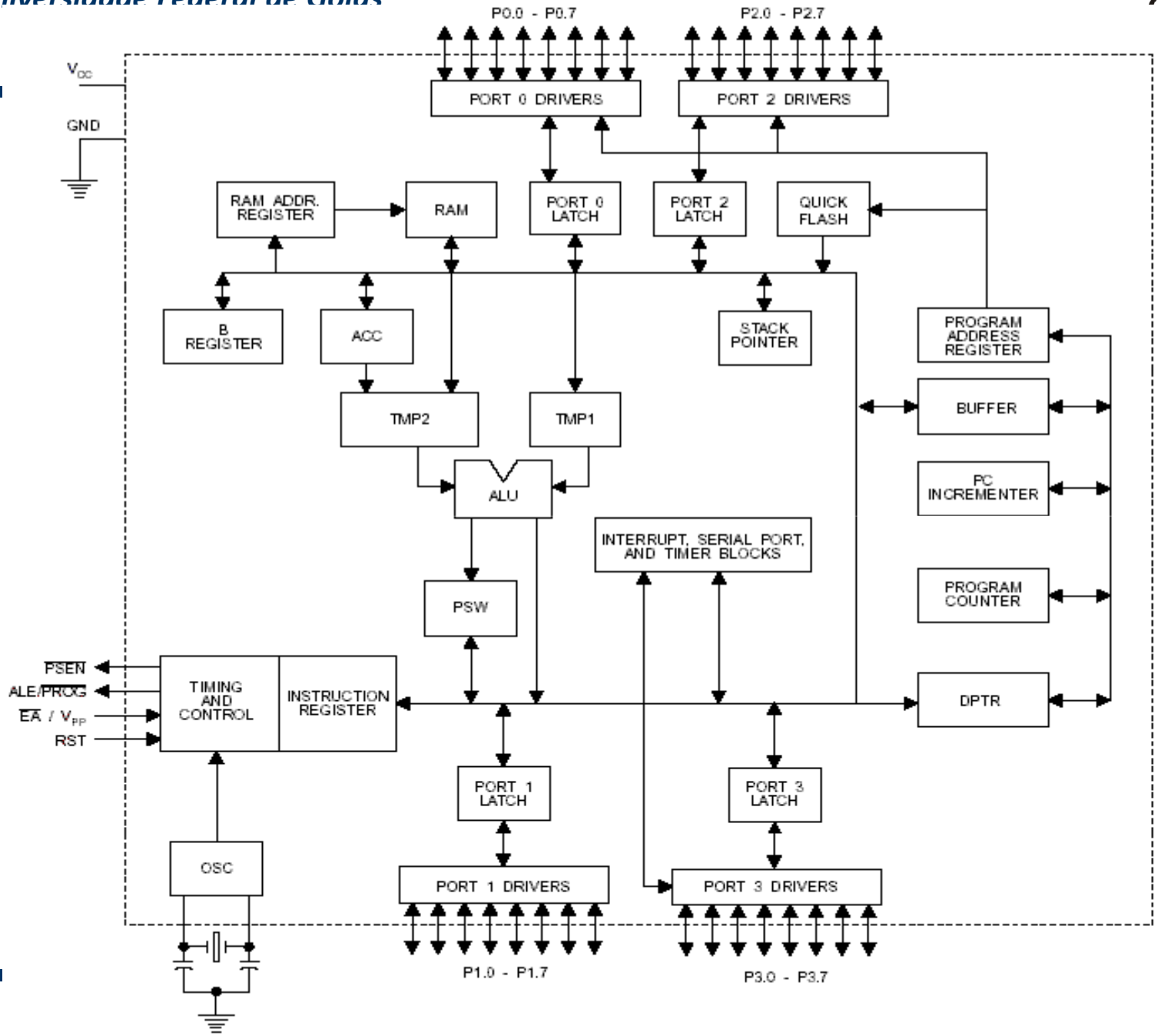
Arquitetura do 8085 (detalhe)



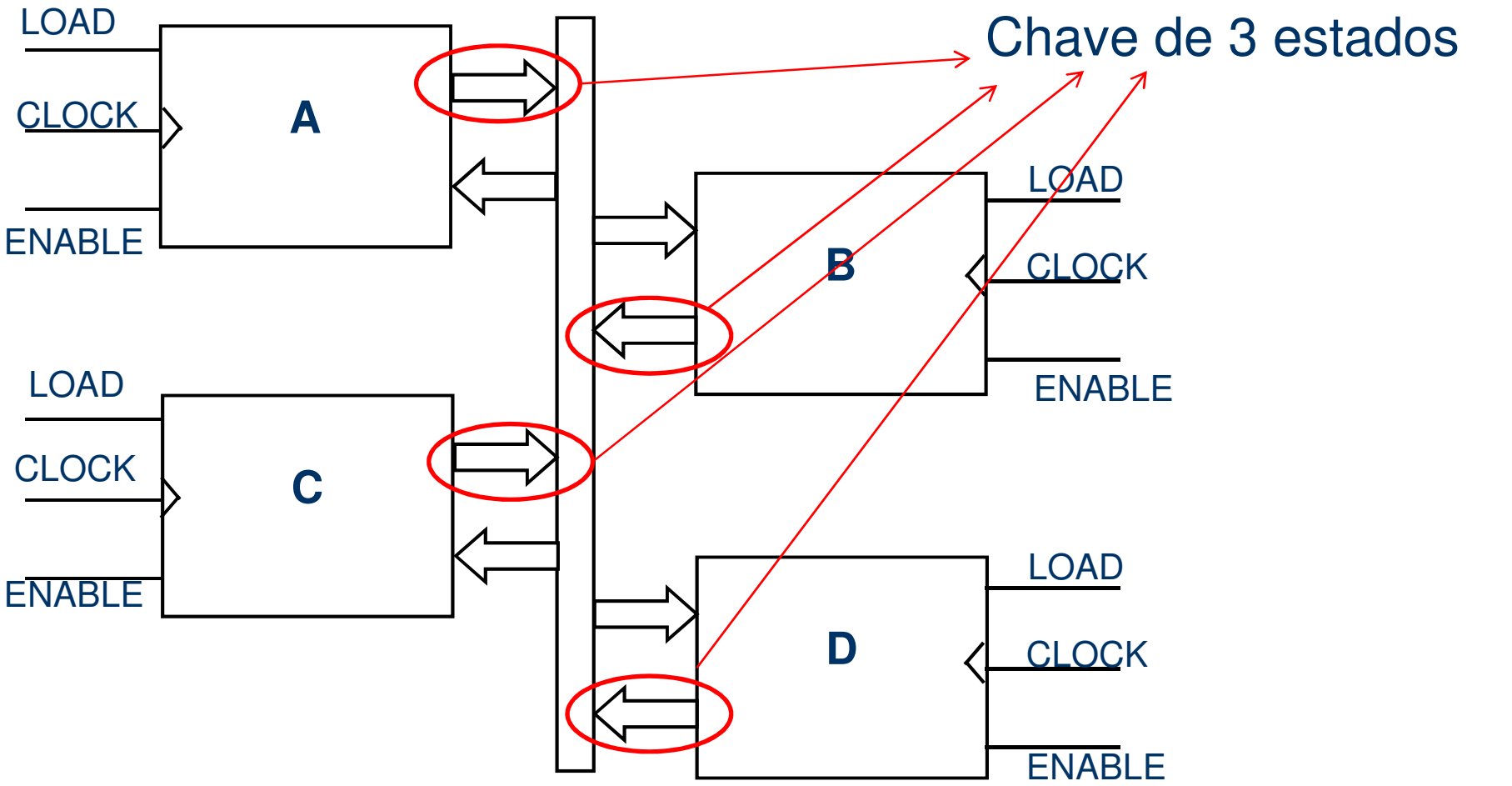
Arquitetura do 8086/88



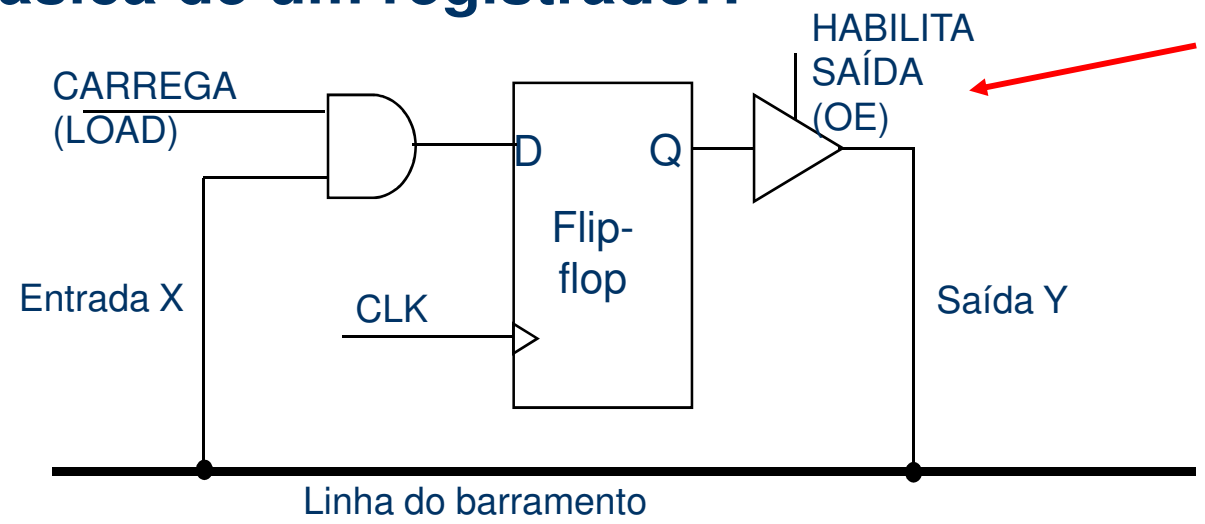
Arquitetura do 8051



Princípio de Funcionamento 8085



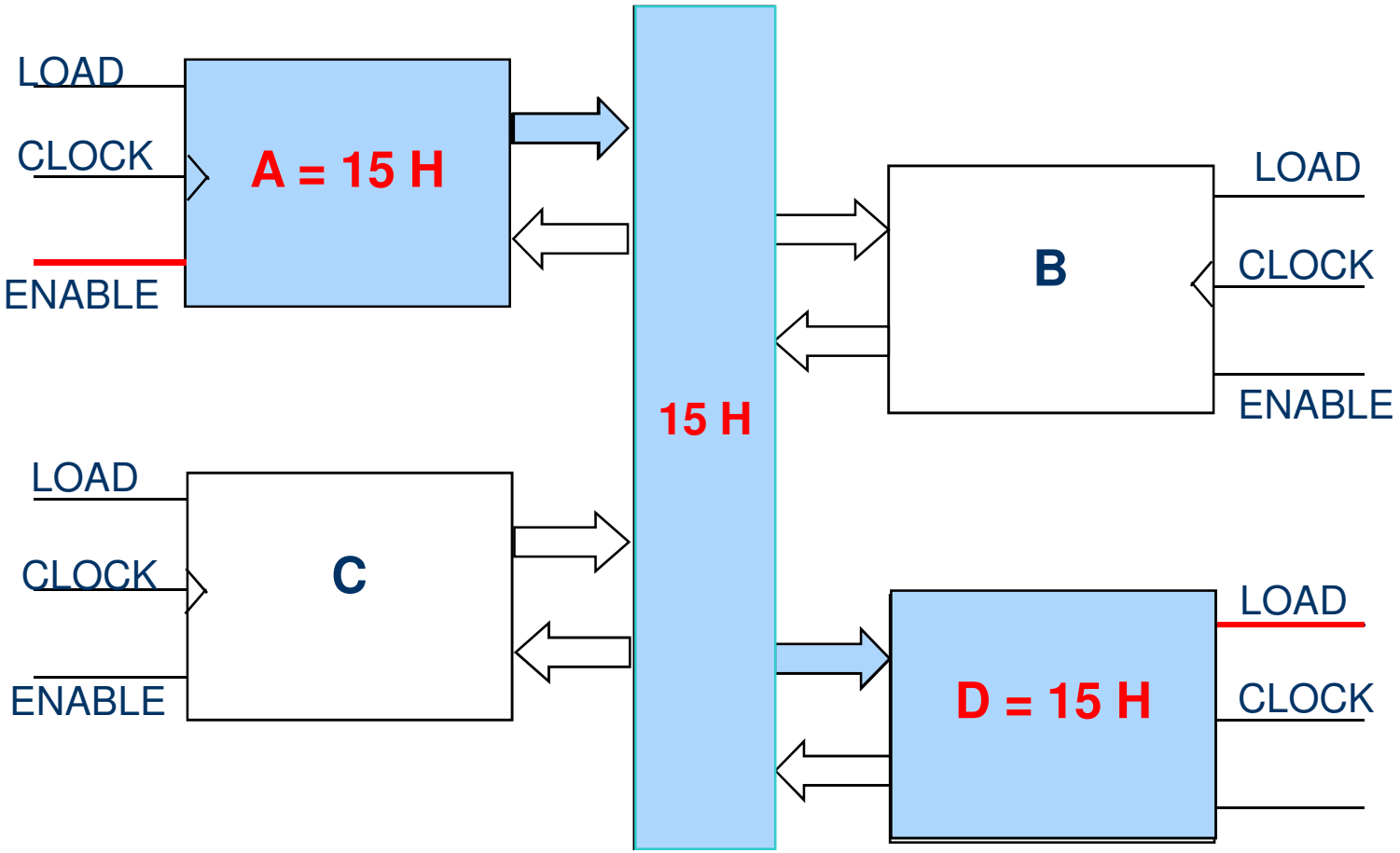
Célula básica de um registrador:



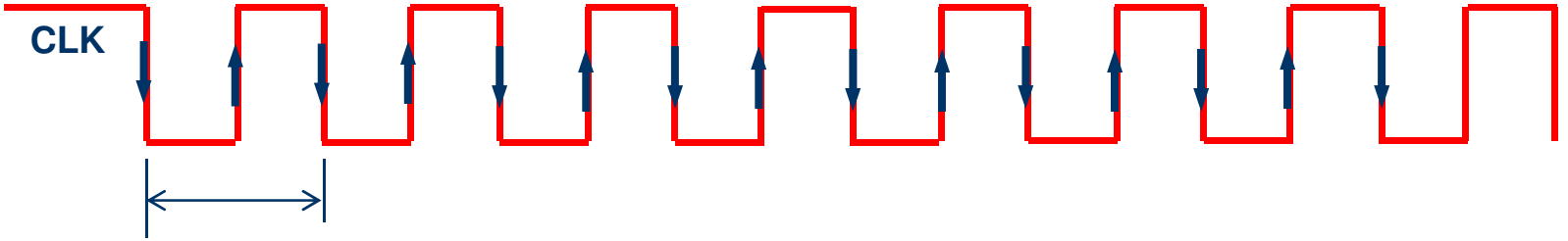
CARREGA REGISTRADOR (LOAD)	HABILITA SAÍDA (OUTPUT ENABLE) (**OE**)	COMENTÁRIO
0	0	Registradores isolados do barramento. Barramento flutuando
0	1	Transfere dados do registrador para o barramento
1	0	Carrega o registrador com os dados do barramento

Princípio de Funcionamento 8085

MOV D,A



Frequência de Clock



Ciclo de Clock

8085A: $f_{cristal}$ = 500 kHz a 3,125 MHz

$$f_{clock} = \frac{f_{cristal}}{2}$$

8085A-2: $f_{cristal}$ = 500 kHz a 5 MHz

Exemplo: Se $f_{cristal}$ = 2 MHz ➔ f_{clock} = 1 MHz

T_{clock} = 1 µs

Ciclos de Clock, de Máquina e de Instrução

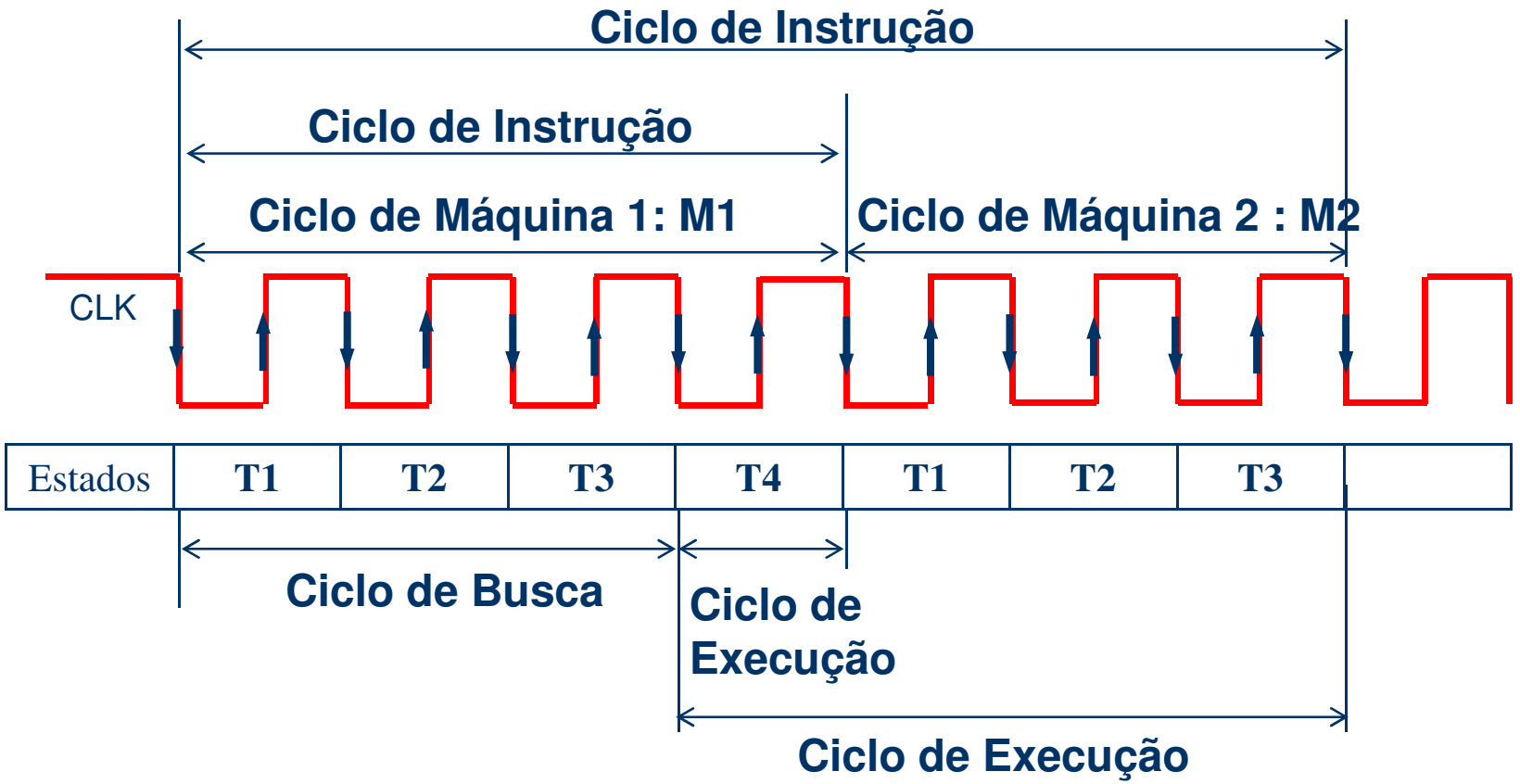
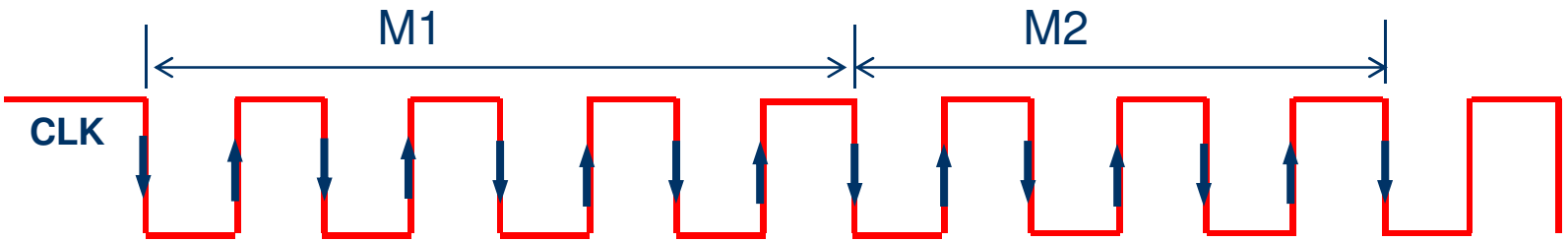


Diagrama de Temporização



Estados	T1	T2	T3	T4	T1	T2	T3	
	PC FORA	PC+1→PC	INST →IR		PC FORA	PC+1→PC	INST →IR	
ALE								
A15-A8		PC H				PC H		
AD7-AD0	PC L		INST		PC L		INST	

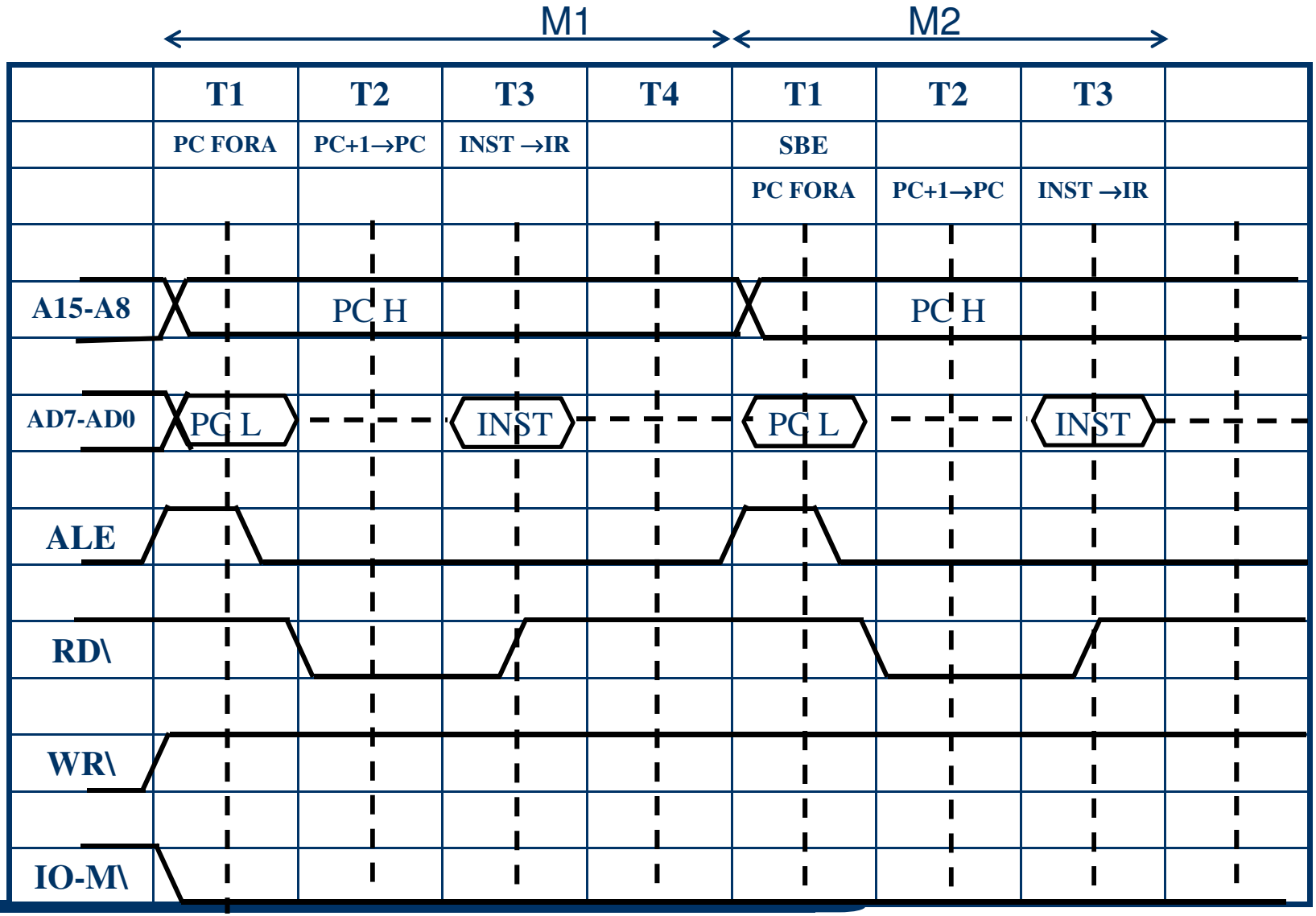
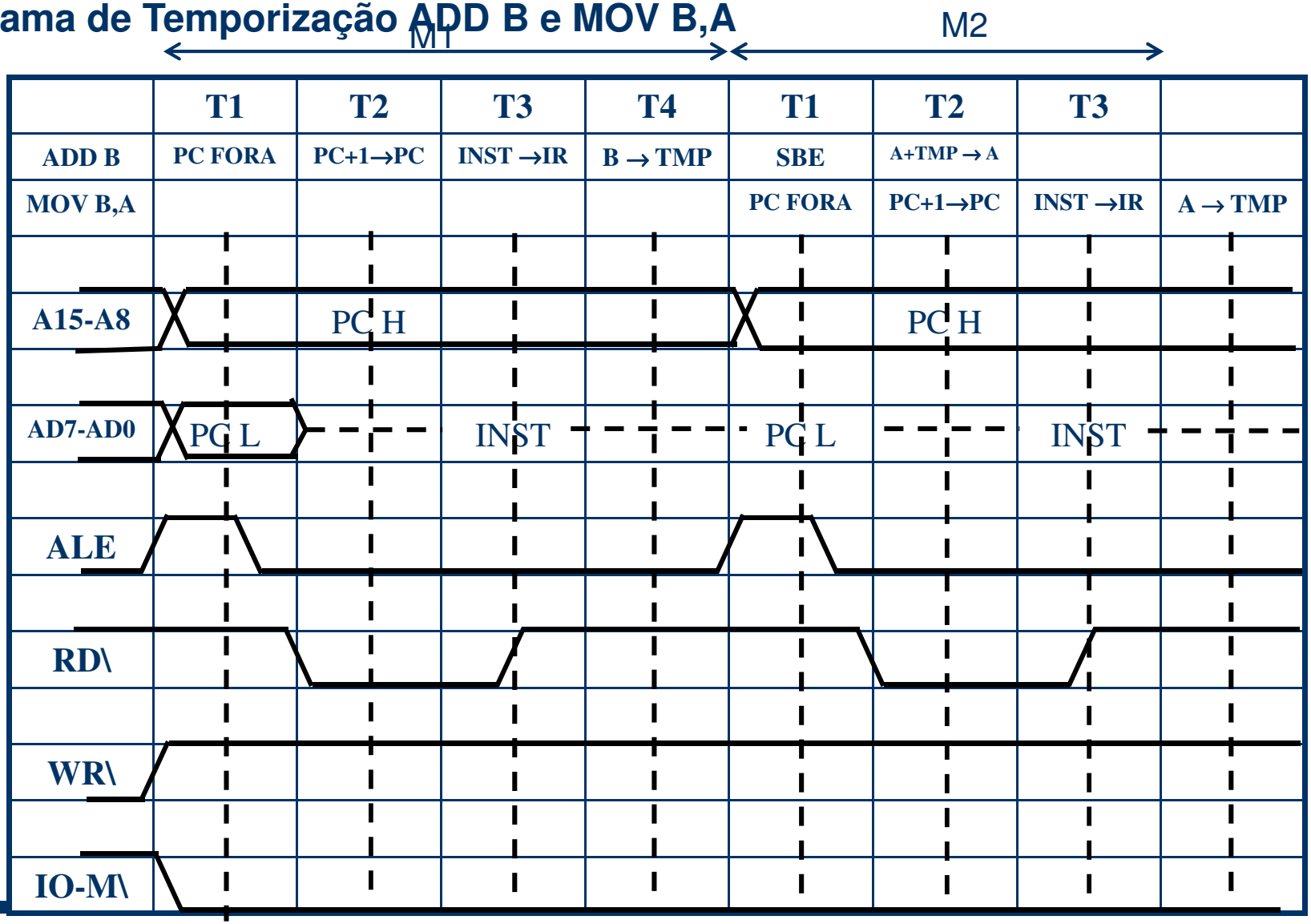

M1
M2
T1
T2
T3
T4
T1
T2
T3
PC FORA
PC+1→PC
INST →IR
SBE
PC FORA
PC+1→PC
INST →IR
A15-A8
PC H
PC H
AD7-AD0
PC L
INST
PC L
INST
ALE
RD\
WR\
IO-M\

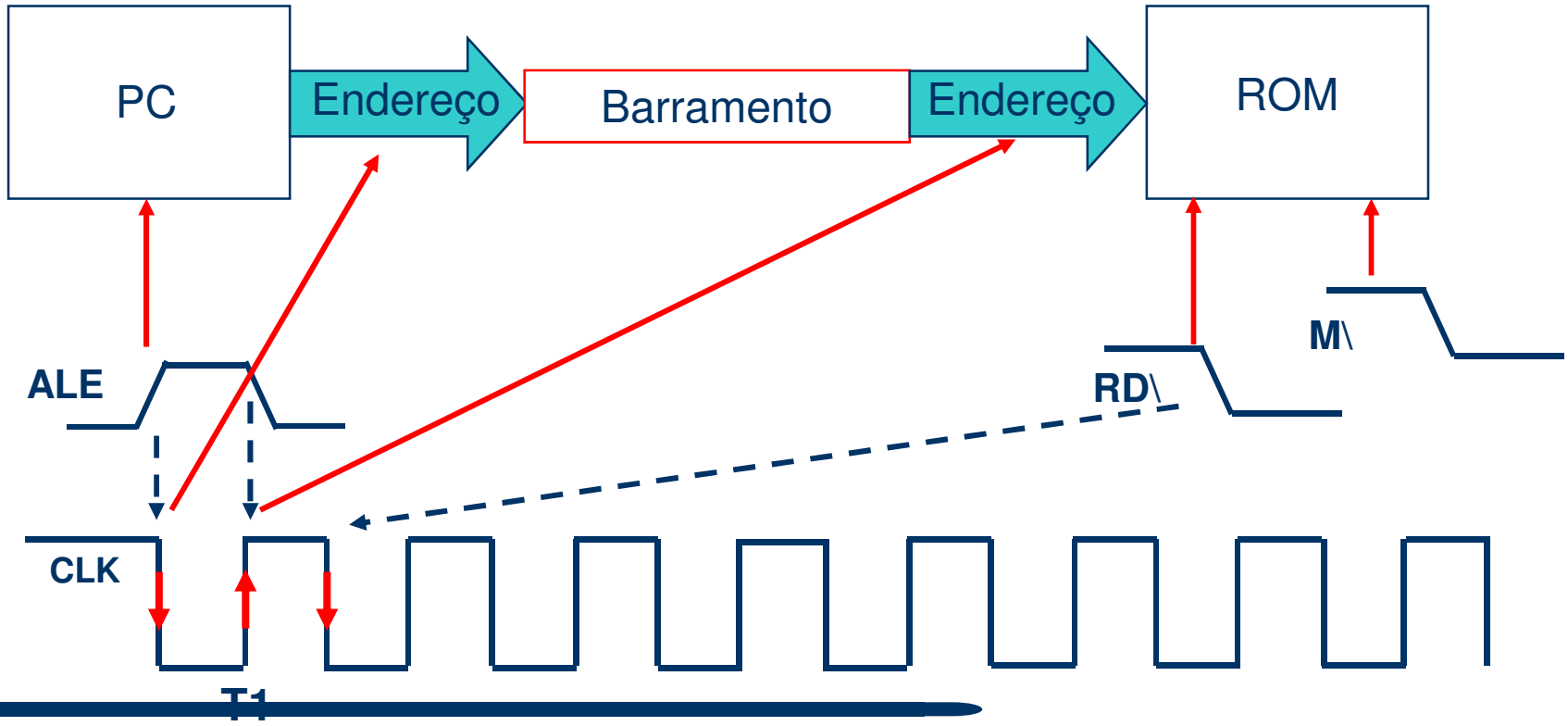
Diagrama de Temporização ADD B e MOV B,A



Princípio de Funcionamento 8085

Ciclo de Busca de ADD B:

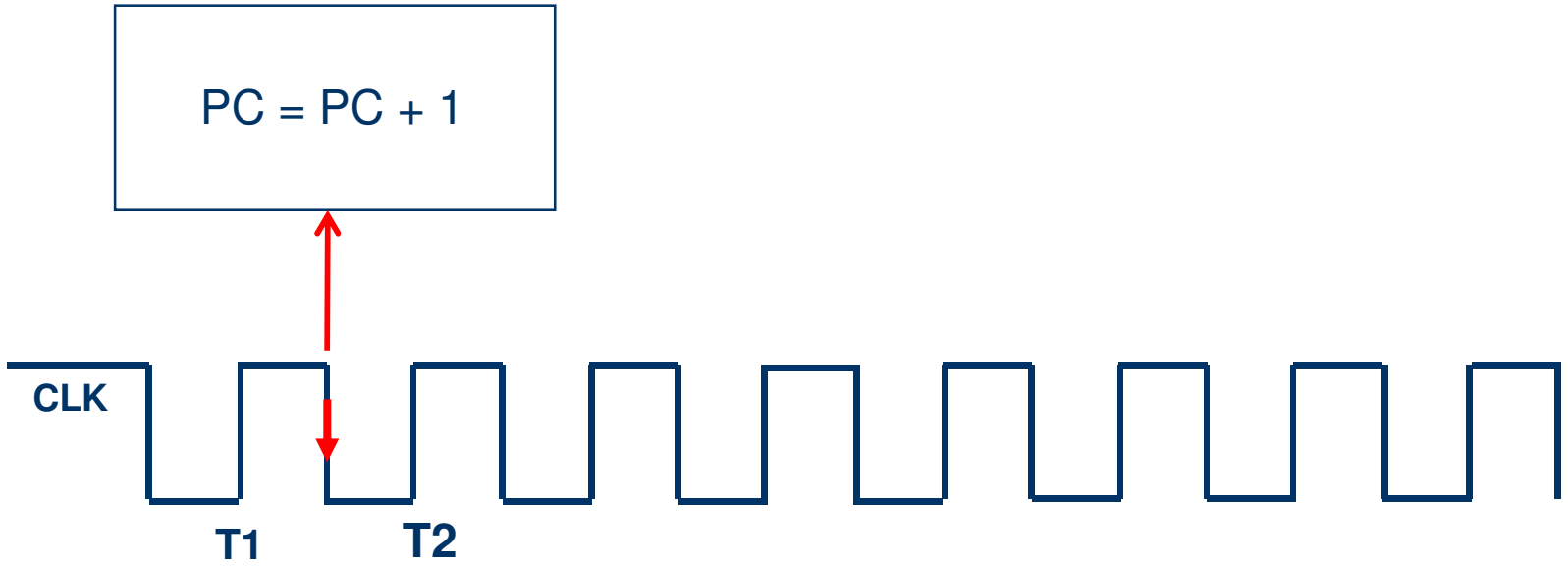
T1: Contador de Programa (PC) é ativado. Endereço atual é colocado no barramento de 16 bits. O sinal ALE é o trigger para a transferência dos endereços de PC para o barramento.



Princípio de Funcionamento 8085

Ciclo de Busca de ADD B:

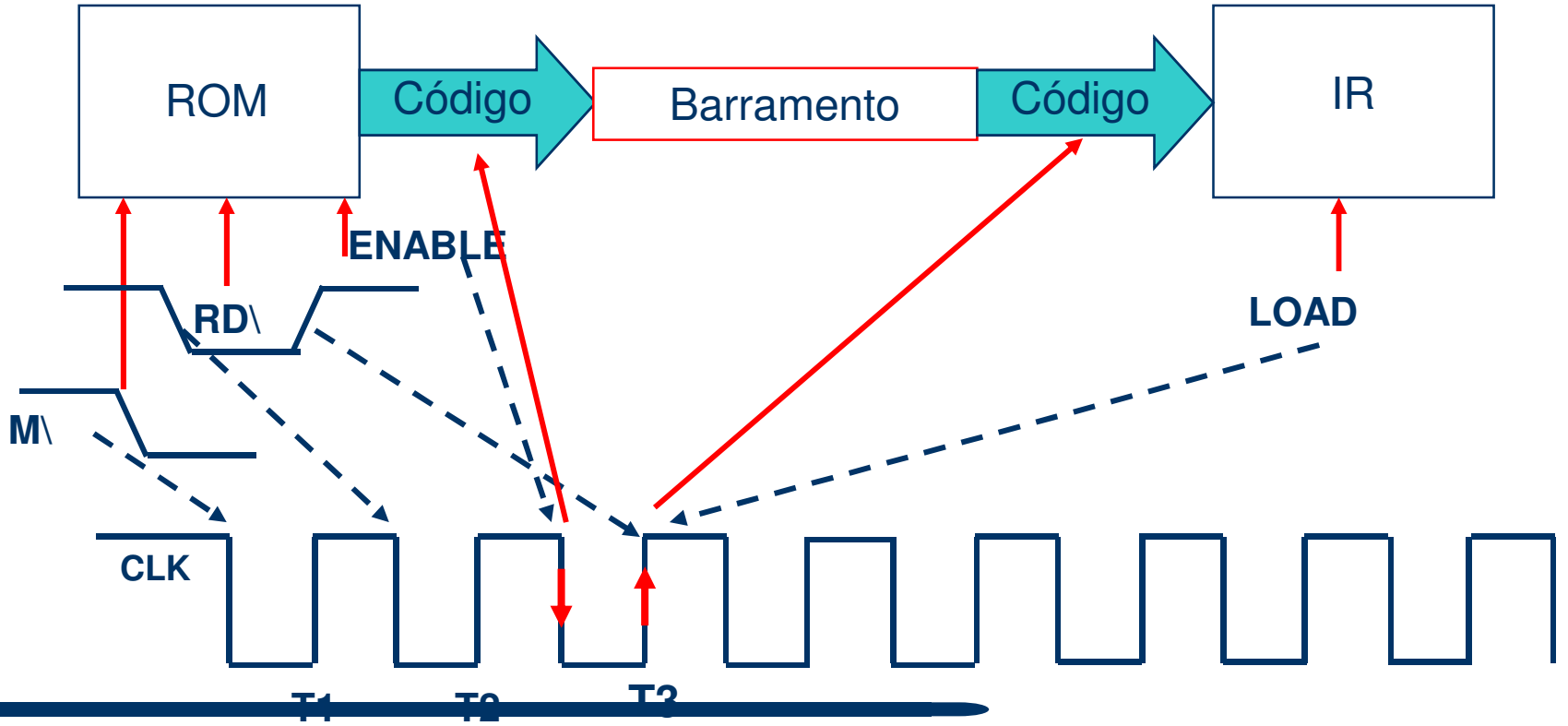
T2: Na descida do sinal de clock o PC é incrementado em 1. Barramento disponível para outras operações.



Princípio de Funcionamento 8085

Ciclo de Busca de ADD B:

T3: Leitura do código da instrução (ROM). O código é transferido para o barramento. Bloco IR (Registrador de Instrução) carrega a instrução vinda da ROM. Instrução é decodificada no Decodificador

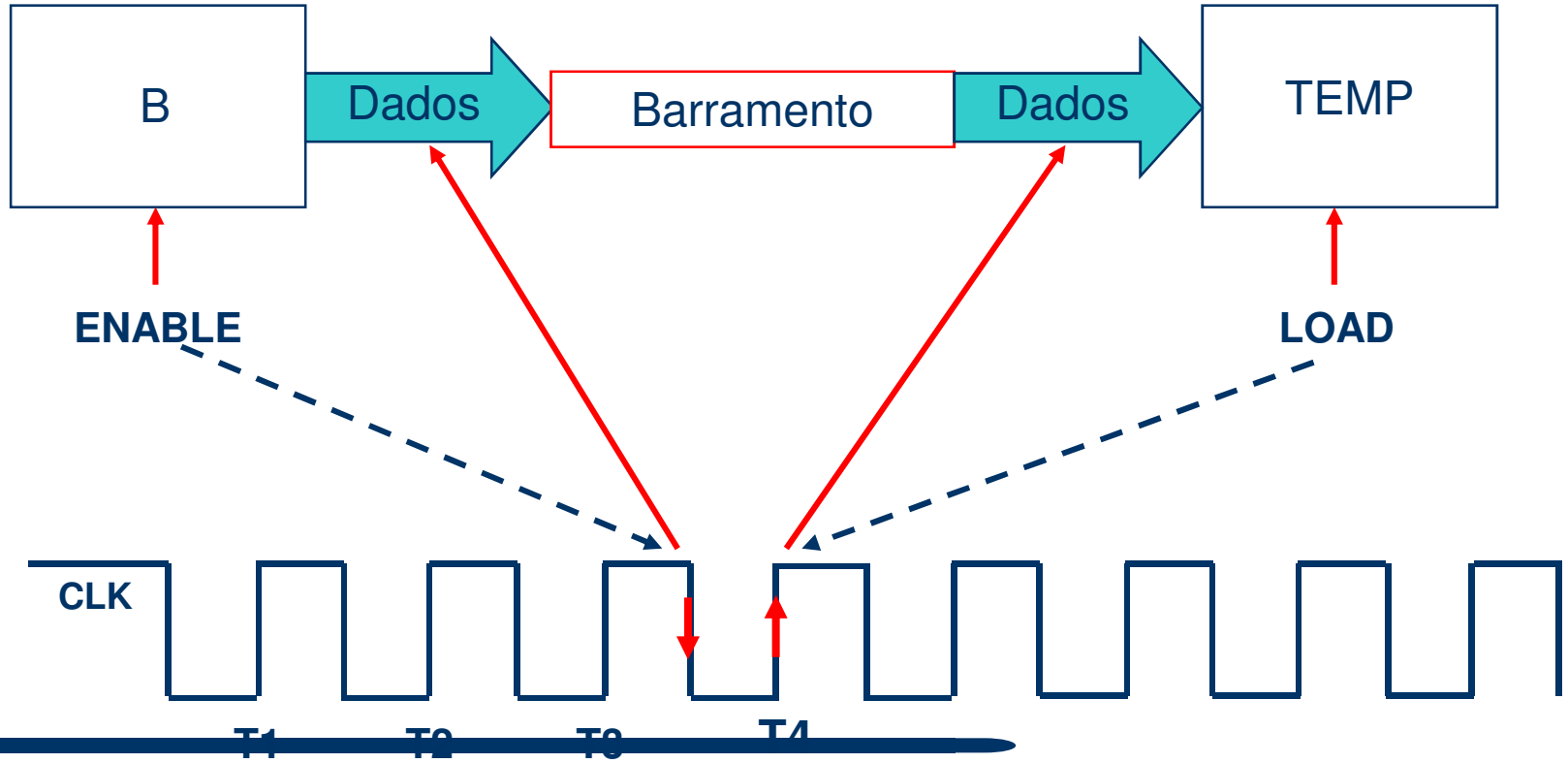


Princípio de Funcionamento 8085

Ciclo de Execução de ADD

T4: Conteúdo de B é transferido para o barramento

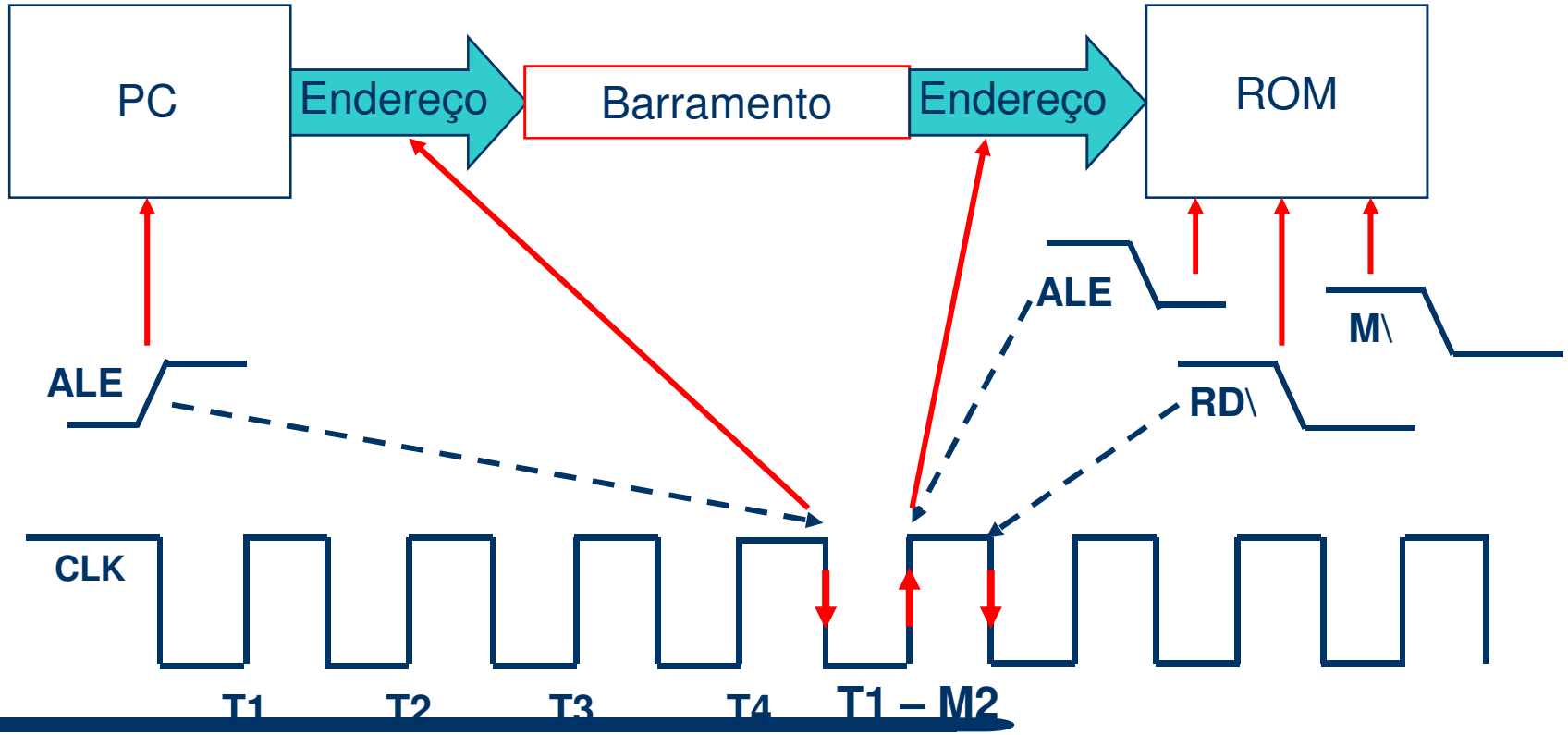
Conteúdo do barramento é transferido para um registrador temporário e, depois para a unidade lógica e aritmética



Princípio de Funcionamento 8085

Ciclo de Busca de MOV B,A e Execução de ADD B:

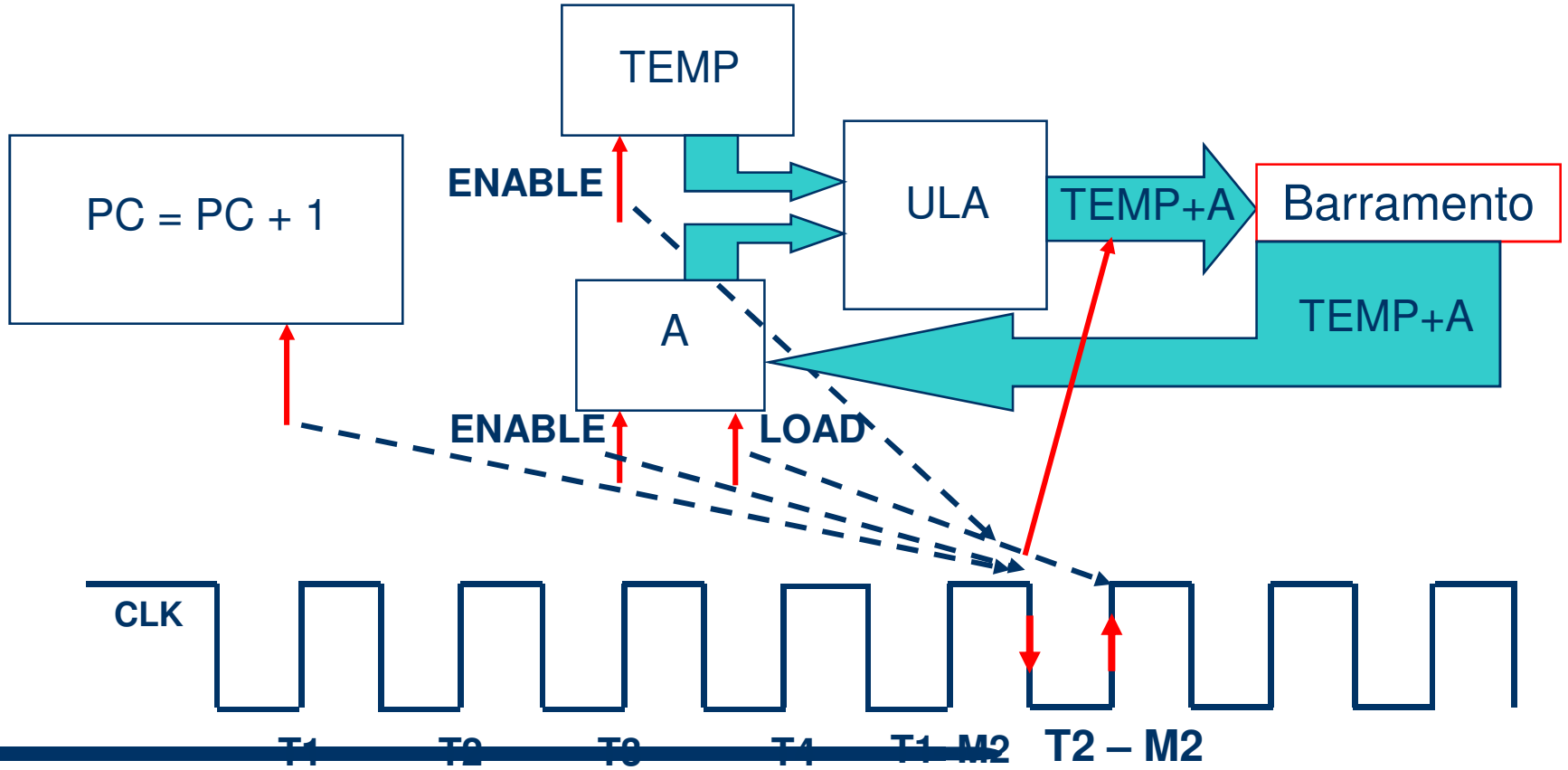
T1: Contador de Programa (PC) é ativado. Endereço atual é colocado no barramento de 16 bits. O sinal ALE é o trigger para a transferência dos endereços de PC para o barramento.



Princípio de Funcionamento 8085

Ciclo de Busca de MOV B,A e Execução de ADD B:

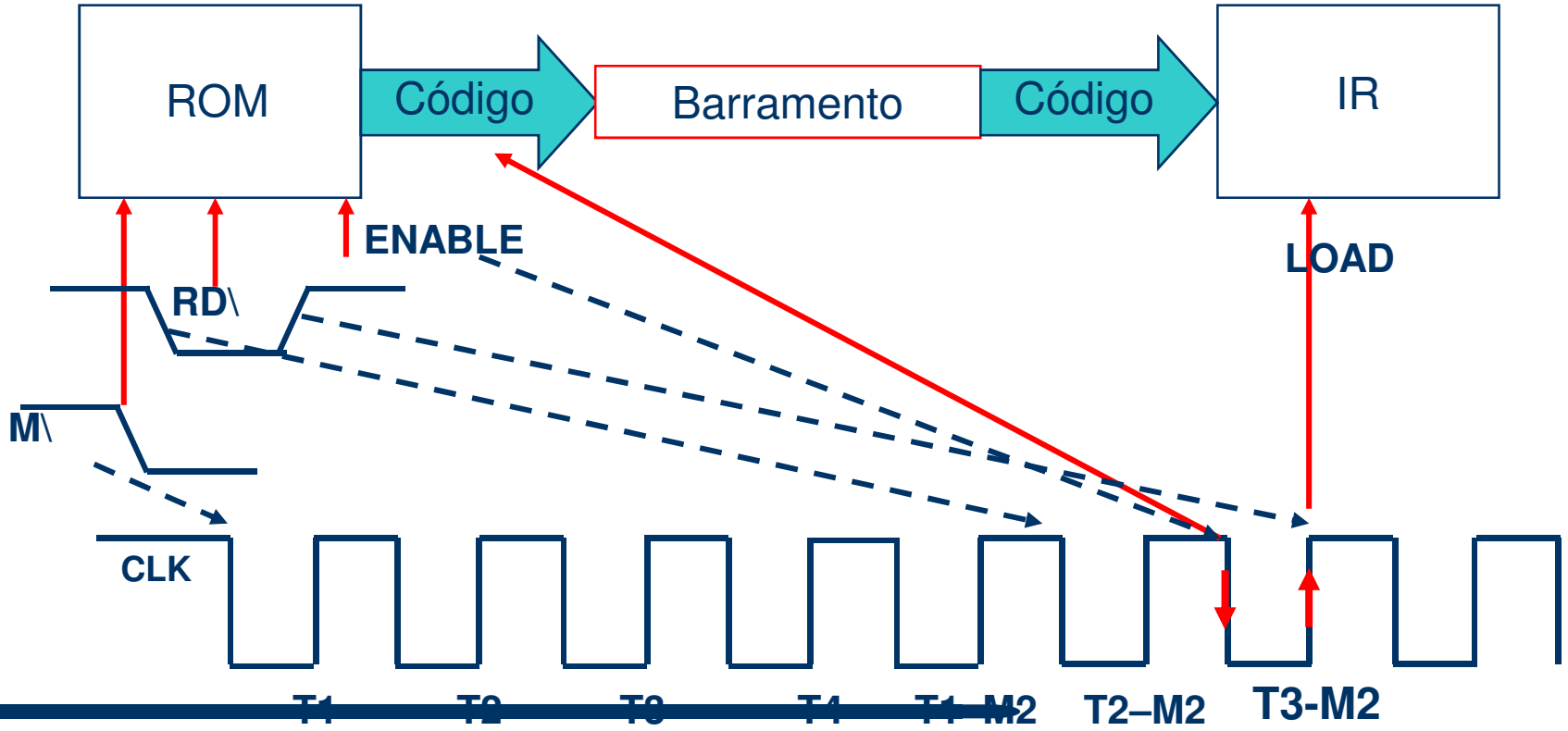
T2: Na descida do sinal de clock o PC é incrementado em 1. Barramento disponível finalizar a instrução ADD B.



Princípio de Funcionamento 8085

Ciclo de Busca de MOV B,A:

T3: Leitura do código da instrução (ROM). O código é transferido para o barramento. Bloco IR (Registrador de Instrução) carrega a instrução vinda da ROM. Instrução é decodificada no Decodificador

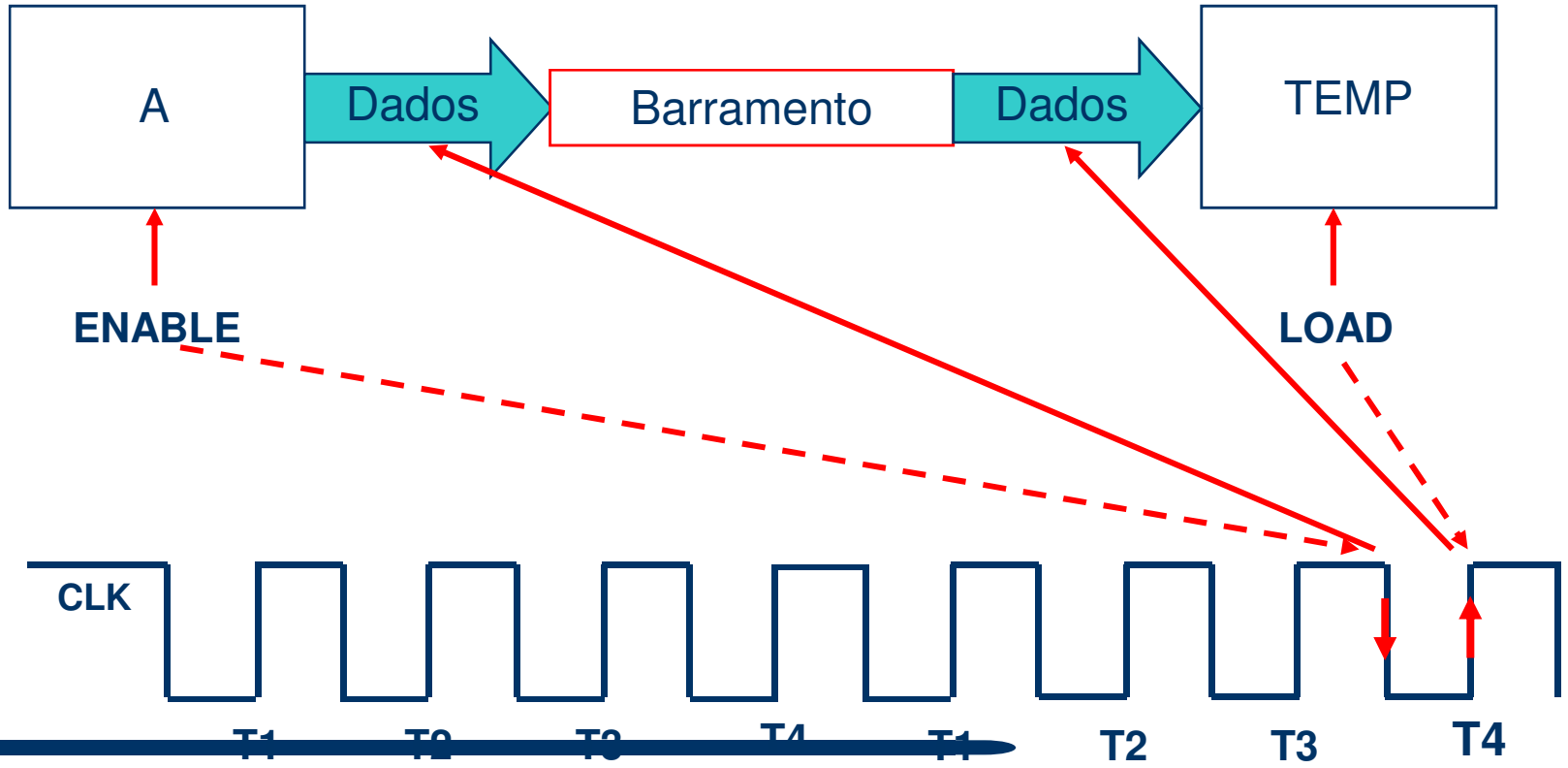


Princípio de Funcionamento 8085

Ciclo de Execução de MOV B,A:

T4: Conteúdo de A é transferido para o barramento

Conteúdo do barramento é transferido para um registrador temporário e, depois para o registrador B, no estado T2 da próxima instrução.



Princípio de Funcionamento 8088/8086:

• A **BIU** coloca o conteúdo do IP (que é somado ao registrador CS) no barramento para efetuar a busca de instrução;

• O registrador IP é incrementado (aponta para a próxima instrução);

• A instrução lida é passada para a fila;

• A **EU** pega a primeira instrução da fila;

• Enquanto a **EU** executa esta instrução a **BIU** faz uma nova busca de instrução para preencher a fila.

• Se a instrução a ser executada pela **EU** for muito demorada a **BIU** preenche toda a fila.

Princípio de Funcionamento 8088/8086:

• Há 2 situações em que não são aproveitadas as instruções contidas na fila. São elas:
Na execução de instruções de desvio. Neste caso a fila é descartada (ou seja, é sobrescrita);

• Quando a instrução faz referência à memória.

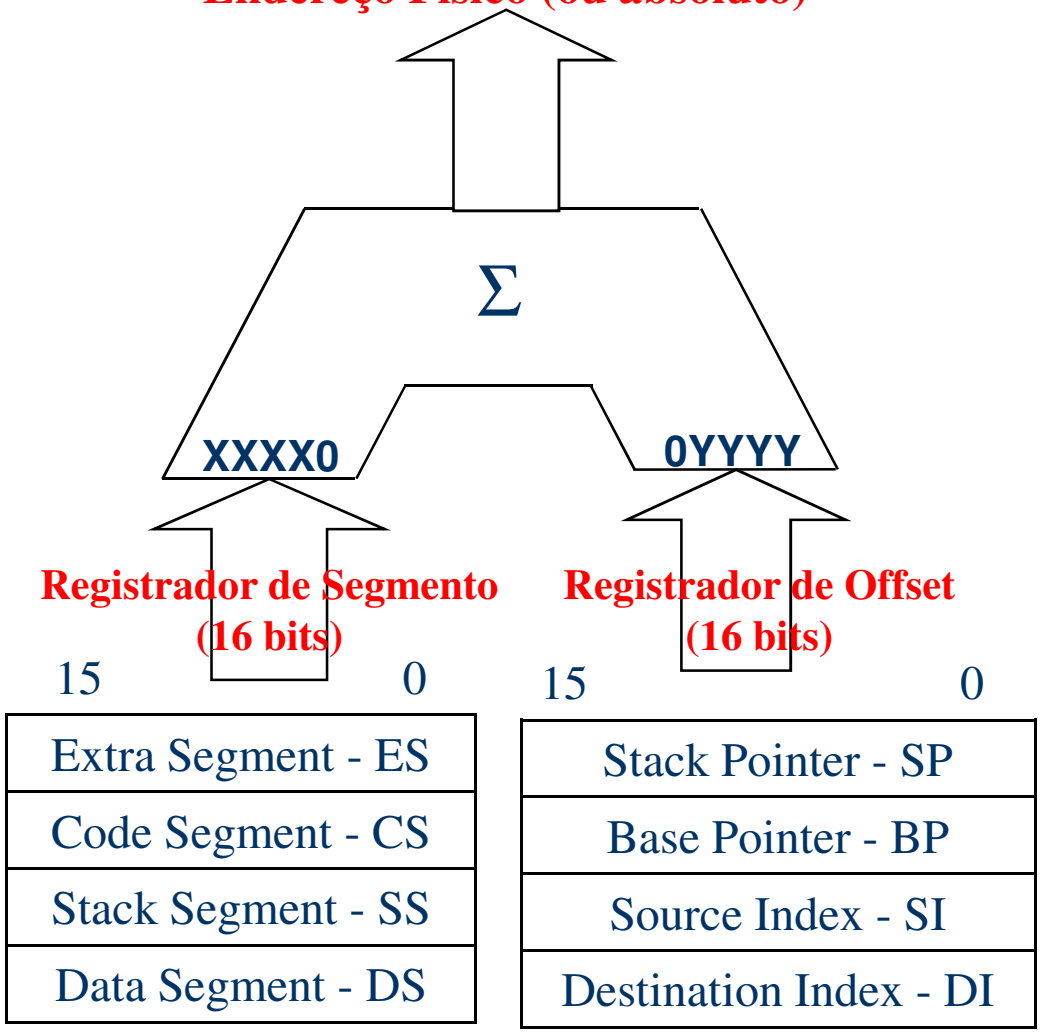
Segmentação

Consiste em combinar 2 registradores de 16 bits para gerar um endereço de memória de 20 bits (2^{20} = 1.048.576 = 1 Mb)

Registrador de Segmento	+	Registrador de Offset

Endereço Físico = (Conteúdo do **Registrador de Segmento**) × **16** +

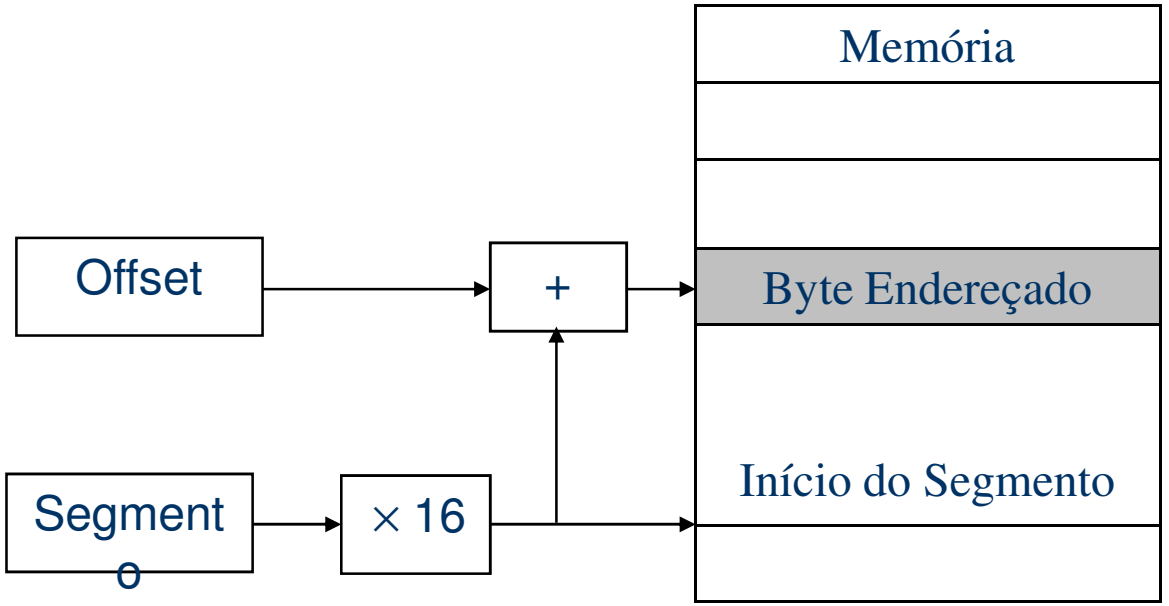
(Conteúdo do **Registrador de Offset**)


Endereço Físico (ou absoluto)
Σ
XXXX0
0YYYY
Registrador de Segmento
(16 bits)
Registrador de Offset
(16 bits)
15
0
15
0
Extra Segment - ES
Code Segment - CS
Stack Segment - SS
Data Segment - DS
Stack Pointer - SP
Base Pointer - BP
Source Index - SI
Destination Index - DI

Vantagens da Utilização de Memória Segmentada

Por haver uma área específica para armazenamento de código e outras áreas para armazenamento de dados, pode-se trabalhar com tipos diferentes de conjuntos de dados.
(por exemplo, em um ambiente multitarefa onde um programa atende várias entradas de dados).

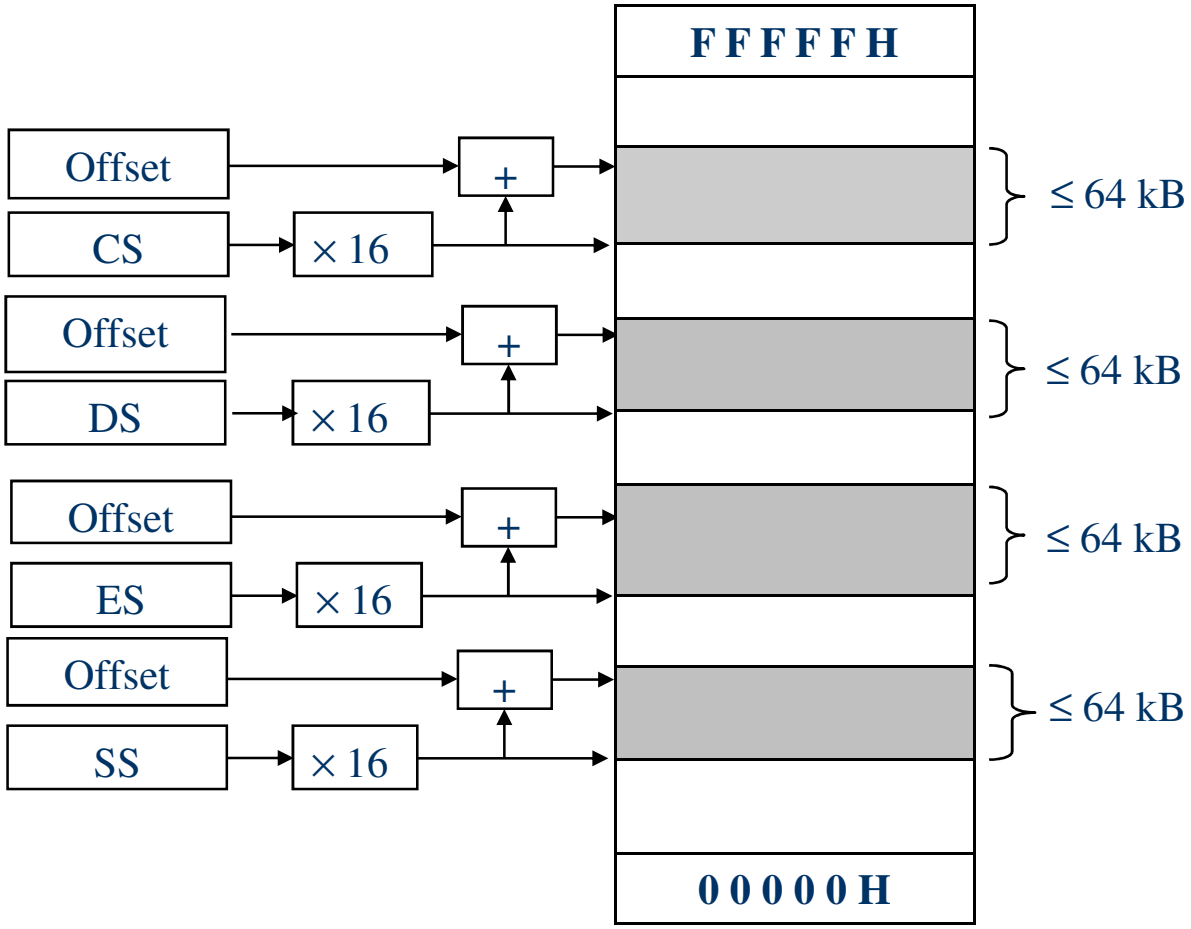
Programas que referenciam endereços lógicos (0000 a FFFF no caso do 8088) podem ser carregados em qualquer espaço (físico) da memória (00000 a FFFFF): possibilita a realocação de programas.



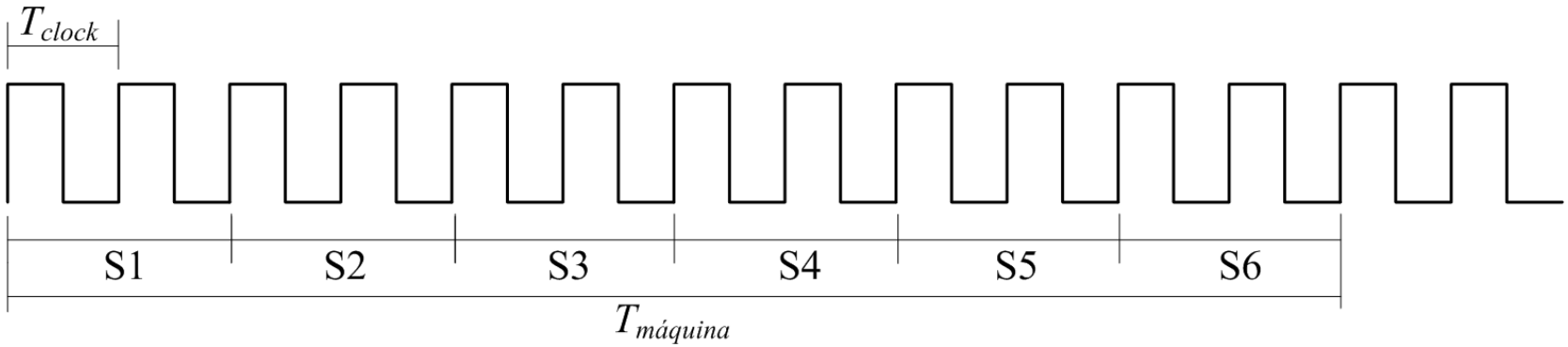
Exemplo 1: Segmento = 2000H; Offset = 2000H
Representação: 2000H:2000H
Endereço Físico = 2000**0H + 0**2000H = 22000H

Exemplo 2: Segmento = 4000H; Offset = 2000H
Representação: 4000H:2000H
Endereço Físico = 4000**0h** + **0**2000h = 42000h

Alocação de diferentes regiões para diferentes conjuntos de dados



Princípio de Funcionamento 8051

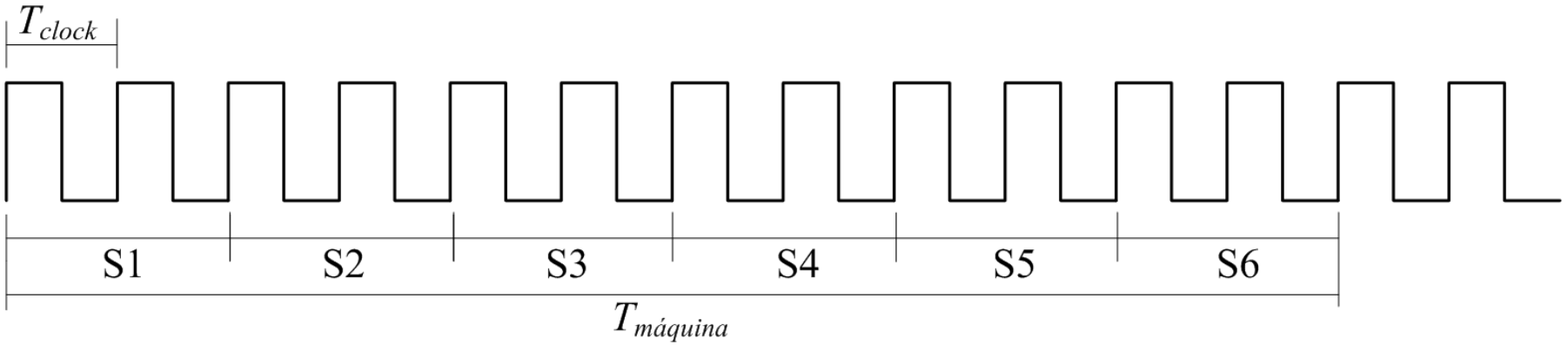


$$T_{clock} = \frac{1}{f_{clock}} \qquad T_{máquina} = 12 \times T_{clock} = \frac{12}{f_{clock}}$$

Se f = 12 MHz → $T_{máquina} = \frac{12}{12MHz} = 1\mu s$

Se f = 11.0592 MHz → $T_{máquina} = \frac{12}{11{,}0592MHz} = 1{,}085\mu s$

Princípio de Funcionamento 8051

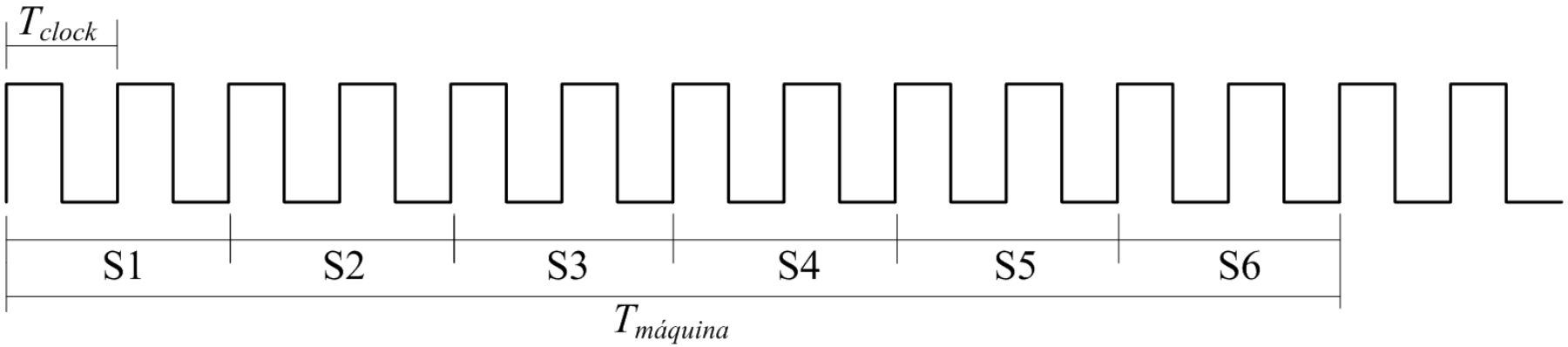


Estado S1: a próxima instrução é buscada na ROM, colocada no barramento principal e encaminhada para o registrador IR.

Estado S2: a instrução é decodificada e o PC é incrementado.

Estado S3: os operandos da instrução são preparados

Princípio de Funcionamento 8051



Estado S4: os operandos são enviados para os registradores temporários TMP1 e TMP2, na entrada da ULA

Estado S5: a ULA executa a instrução

Estado S6: o resultado da ULA é colocado no barramento principal e encaminhado para o registrador final.

Formato das Instruções

Formato das Instruções do 8085

Tipo de instrução	Características	Exemplos
1 byte	O byte da instrução é o próprio Opcode (código de operação)	MOV A,C ADD B RLC DCR C
2 bytes	O primeiro byte é o Opcode e o segundo byte é o Dado de 8 bits necessário para a instrução	MVI A,35H ADI 05H ORI 01H
3 bytes	O primeiro byte é o Opcode; o segundo e o terceiro bytes correspondem a um dado de 16 bits.	LDA 2030H STA 2040H LXI H,2080H

Formato das Instruções do 8085

Instrução	MOV A,C
Opcode	79 H

Instrução	ADD B
Opcode	80 H

Instrução	RLC
Opcode	07 H

Instrução	MVI A,35H
Opcode	3E H
Dado	35 H

Instrução	ADI 05H
Opcode	C6 H
Dado	05 H

Instrução	ORI 01H
Opcode	F6 H
Dado	01 H

Instrução	LDA 2030H
Opcode	3A H
Dado L	30 H
Dado H	20 H

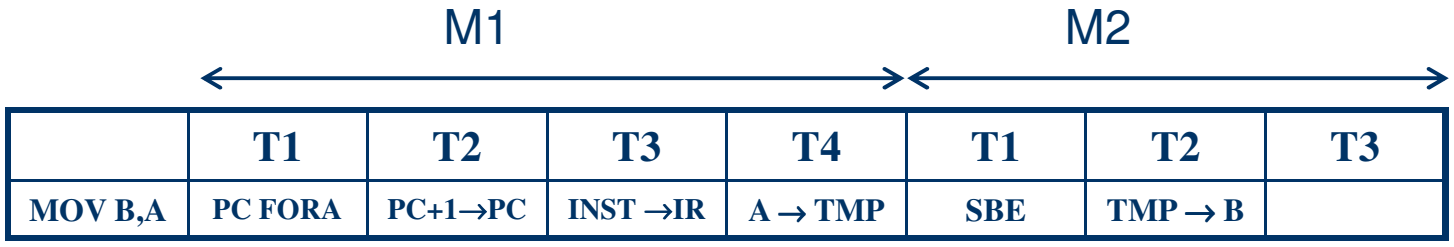
Instrução	STA 2050H
Opcode	32H
Dado L	50 H
Dado H	20 H

Instrução	LXI H,2080H
Opcode	21H
Dado L	80 H
Dado H	20 H

Diagrama de Temporização (Instrução de 1 byte)

	T1	T2	T3	T4	T1	T2	T3
ADD B	PC FORA	PC+1→PC	INST →IR	B → TMP	SBE	A+TMP → A	

ADD B ➔ Adiciona o conteúdo de B ao conteúdo do Acumulador A



	T1	T2	T3	T4	T1	T2	T3
MOV B,A	PC FORA	PC+1→PC	INST →IR	A → TMP	SBE	TMP → B	

MOV B,A ➔ Copia em B o conteúdo do registrador A

Diagrama de Temporização (Instrução de 2 bytes)

M1 M2

	T1	T2	T3	T4	T1	T2	T3
MVI reg, byte	PC FOR A	PC+1→PC	INST →IR	X	PC FOR A	PC+1→PC	byte→reg

MVI reg,dado ➔ Carrega "reg" com o valor "dado"

M1 M2

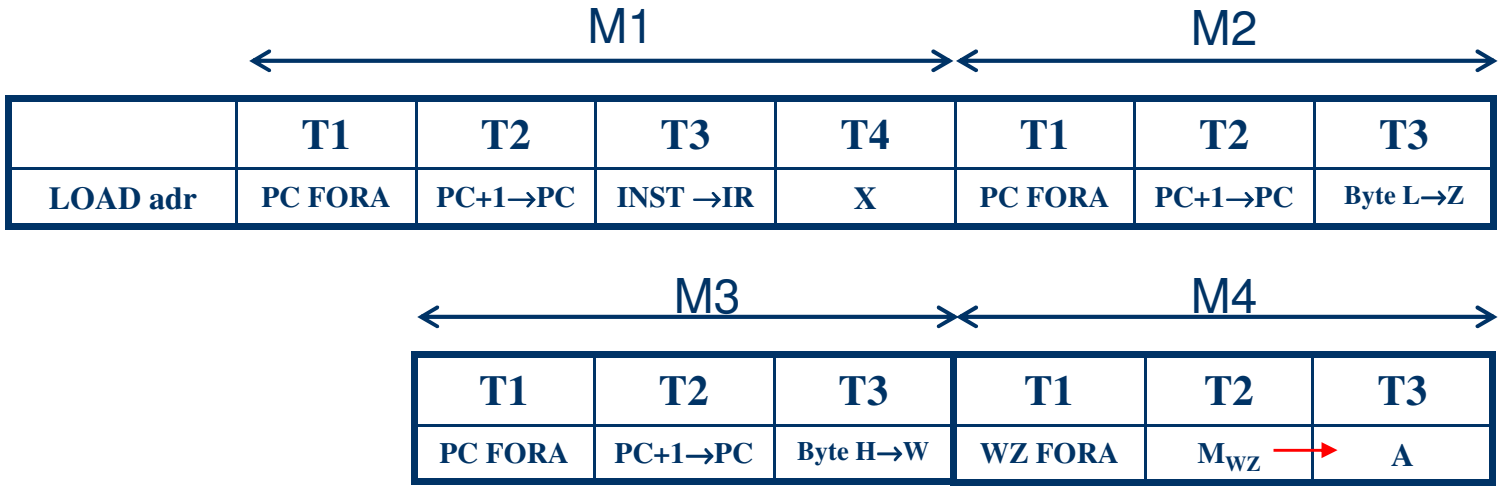
	T1	T2	T3	T4	T1	T2	T3
ADI byte	PC FORA	PC+1→PC	INST →IR	X	PC FORA	PC+1→PC	byte→TMP

ADI dado➔ Adiciona o "dado" ao conteúdo do Acumulador A

M3

T1	T2	T3
SBE	A+TMP→ A	

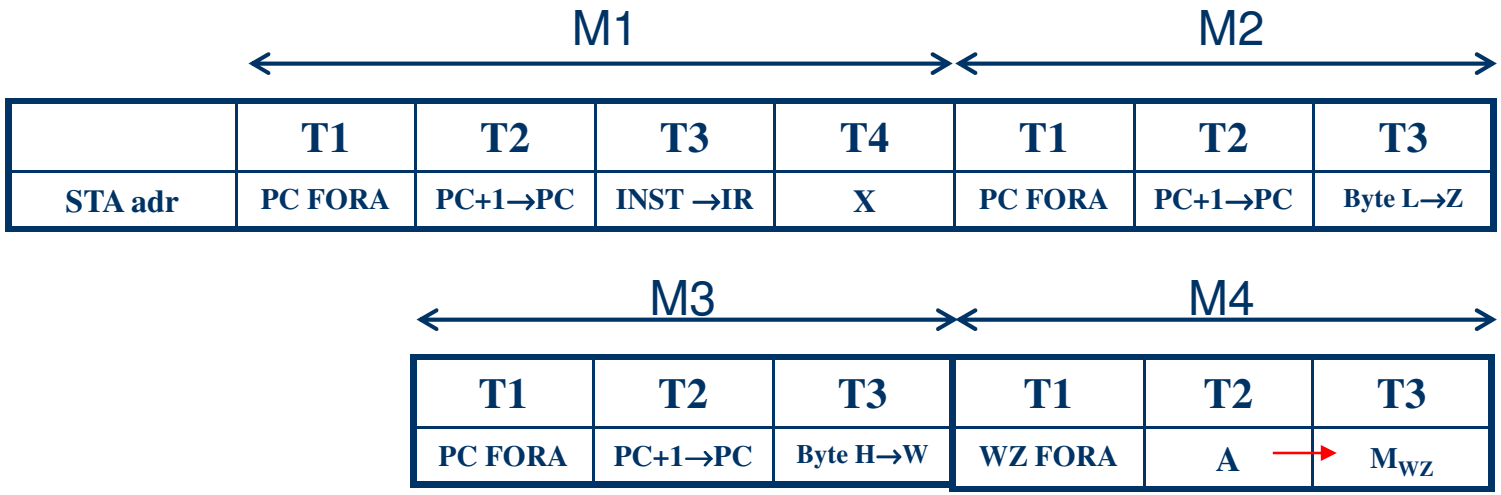
Diagrama de Temporização (Instrução de 3 bytes)



LOAD adr ➔ Carrega no Acumulador o conteúdo do endereço “adr”

Instrução de 3 bytes

Diagrama de Temporização (Instrução de 3 bytes)



STA adr ➔ Transfere conteúdo do Acumulador para o endereço “adr”

Formato das Instruções de transferência de dados do 8086/88

Byte 1			Byte 2	Byte 3	Byte 4
7 2	1	0	7 0	7 0	7 0
opcode	*s*	*w*	***postbyte***	***Dados***	***Dados (se sw=0 1)***

Ou

7	6	5	4	3	2	1	0	
Opcode (código da operação)						*s*	*w*	Byte 1
postbyte								Byte 2
Dados								Byte 3
Dados (se s w = 0 1)								Byte 4

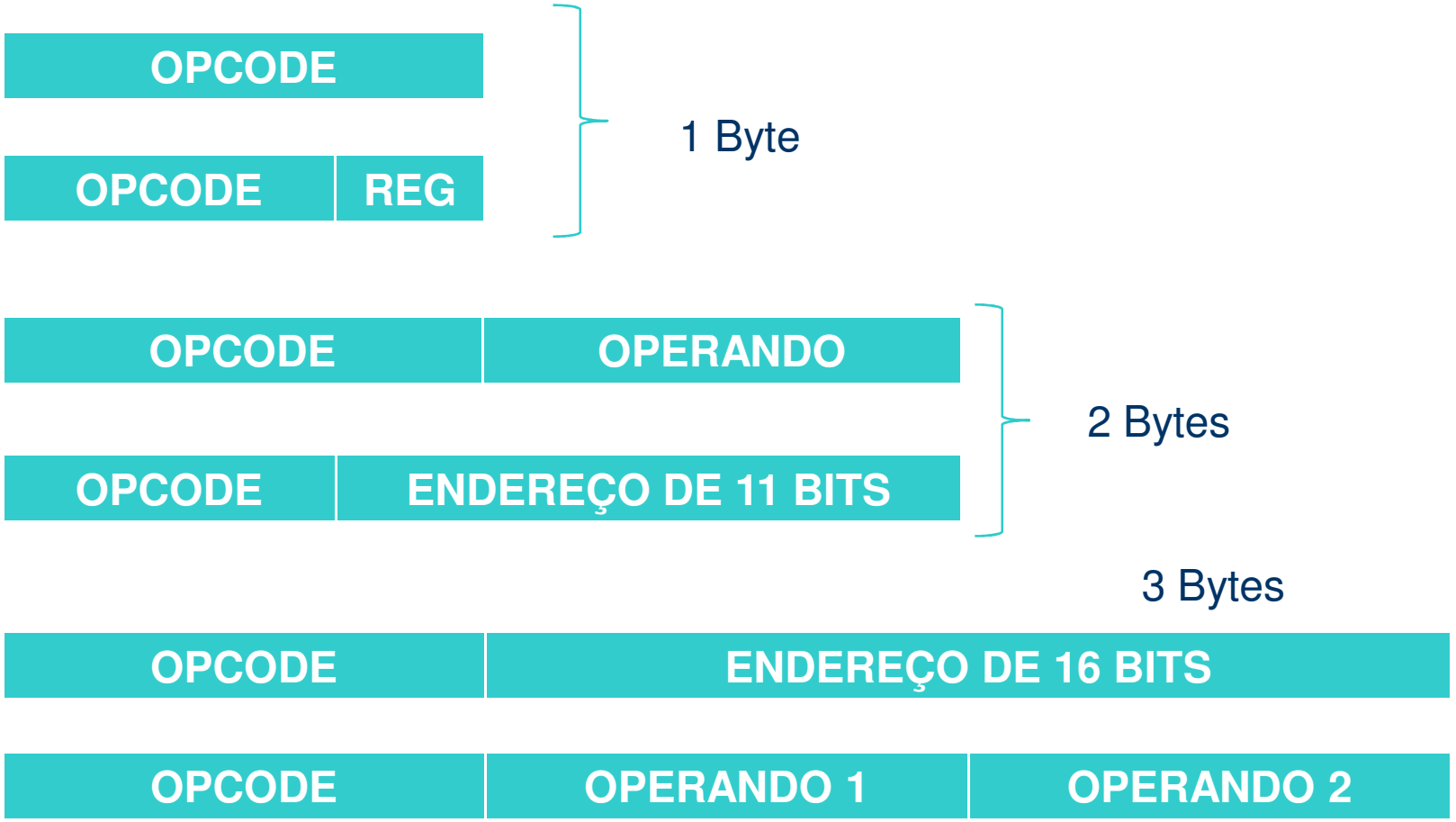
s	*w*	Efeito
0	0	Instrução manipula byte
0	1	Instrução manipula word

postbyte:

7	6	5	4	3	2	1	0
mod		reg			r/m		

Se ***mod*** = 1 1 ⇒ o operando da instrução é um registrador, que é identificado em "**r/m**"
Se a instrução envolver dois registradores o campo "**reg**" identifica o segundo registrador.

Formato das Instruções do 8051



Formato das Instruções do 8051 – Exemplos

OPCODE

Instrução	RLC A
Codificação	0 0 1 1:0 0 1 1 (33H)

Instrução	CLR A
Codificação	1 1 1 0:0 1 0 0 (E4H)

Instrução	CLR C
Codificação	1 1 0 0:0 0 1 1 (C3H)

Formato das Instruções do 8051 – Exemplos

OPCODE	REG

Instrução	ADD A,Rn
Codificação	0 0 1 0:1 r r r (2_H)

Instrução	ADD A,R0
Codificação	0 0 1 0:1 0 0 0 (28H)

Instrução	ADD A,R1
Codificação	0 0 1 0:1 0 0 1 (29H)

Instrução	MOV A,Rn
Codificação	1 1 1 0:1 r r r (E_H)

Instrução	MOV A,R0
Codificação	1 1 1 0:1 0 0 0 (E8H)

Formato das Instruções do 8051 – Exemplos

OPCODE	OPERANDO

Instrução	ADD A,#Dado	
Codificação	0 0 1 0:0 1 0 0	d7 d6 d5 d4 d3 d2 d1 d0 (24 XX) H

Instrução	ADD A,#35H	
Codificação	0 0 1 0:0 1 0 0	0 0 1 1:0 1 0 1 (24 35)H

Instrução	MOV A,#Dado	
Codificação	0 1 1 1:0 1 0 0	d7 d6 d5 d4 d3 d2 d1 d0 (74 XX)H

Instrução	MOV A,#35H	
Codificação	0 1 1 1:0 1 0 0	0 0 1 1:0 1 0 1 (74 35)H

Formato das Instruções do 8051 – Exemplos

OPCODE	ENDEREÇO DE 11 BITS

Instrução	AJMP End. 11 bits	
Codificação	a10 a9 a8 0 : 0 0 0 1	d7 d6 d5 d4 d3 d2 d1 d0 (X1 XX) H

11 bits de endereço → 2^{11} = 2048 → 0 a 2047 ou 000 H a 7FF H

620 H = 0 1 1 0 0 0 1 0 0 0 0 0 b → a10 = a9 = 1 e a8 = 0 →

→ a10 a9 a8 0: 0 0 0 1 → 1 1 0 0 : 0 0 0 1 = C1 H

Instrução	AJMP 620H	
Codificação	1 1 0 0 : 0 0 0 1	0 0 1 0 : 0 0 0 0 (C1 20)H

Formato das Instruções do 8051 – Exemplos

OPCODE	ENDEREÇO DE 16 BITS

Instrução	LCALL End. 16 bits
Codificação	0 0 0 1:0 0 1 0 a15 a14 a13 a12:a11 a10 a9 a8 a7 a6 a5 a4:a3 a2 a1 a0

0100 H = 0 0 0 0 : 0 0 0 1 0 0 0 0 : 0 0 0 0 b

Instrução	LCALL 0100 H
Codificação	0 0 0 1:0 0 1 0 0 0 0 0 : 0 0 0 1 0 0 0 0 : 0 0 0 0 → (12 01 00) H

Formato das Instruções do 8051 – Exemplos

OPCODE	ENDEREÇO DE 16 BITS

Instrução	LJMP End. 16 bits
Codificação	0 0 0 0 : 0 0 1 0 a15 a14 a13 a12 : a11 a10 a9 a8 a7 a6 a5 a4 : a3 a2 a1 a0

0120 H = 0 0 0 0 : 0 0 0 1 0 0 1 0 : 0 0 0 0 b

Instrução	LJMP 0120 H
Codificação	0 0 0 0 : 0 0 1 0 0 0 0 0 : 0 0 0 1 0 0 1 0 : 0 0 0 0 → (02 01 20) H

LJMP 80 H X SJMP 80 H:

02 00 80 X 80 XX H (XX = 80 H – Posição da instrução seguinte a SJMP)

Formato das Instruções do 8051 – Exemplos

OPCODE	OPERANDO 1	OPERANDO 2

Instrução	DJNZ Direto, Relativo8		
Codificação	1 1 0 1 : 0 1 0 1	a7 a6 a5 a4 : a3 a2 a1 a0	r7 r6 r5 r4 : r3 r2 r1 r0

Instrução	DJNZ 01H, $		
Codificação	1 1 0 1 : 0 1 0 1	0 0 0 0 : 0 0 0 1	1 1 1 1 : 1 1 0 1

DJNZ 01H,$ → D5 01 FD H

Formato das Instruções do PIC 16F628

Todas as instruções tem 14 bits

Tipo 1: Operações com registradores orientadas por byte

13					8	7	6	5	4	3	2	1	0
OPCODE						d	f (Endereço do registrador)						

OPCODE → Código da operação

d = 0 → o destino do resultado é o registrador W

d = 1 → o destino do resultado é o registrador f

f → Registrador de endereços de 7 bits (operando)

Formato das Instruções do PIC 16F628

Tipo 2: Operações com registradores orientadas por bit

13			10		8	7	6	5	4	3	2	1	0
OPCODE					b		f (Endereço do registrador)						

OPCODE → Código da operação

b → Endereço de 3 bits

f → Registrador de endereços de 7 bits (operando)

Formato das Instruções do PIC 16F628

Tipo 3: Operações de controle e literal

13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPCODE						k (literal)							

OPCODE → Código da operação

k → Indicam um valor constante ou literal

Formato das Instruções do PIC 16F628

Tipo 4: Operações de desvio

13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPCODE			k (literal)										

OPCODE → Código da operação

k → Indicam um valor constante ou literal de 11 bits

Modos de Endereçamento e Grupos de Instruções

Modos de Endereçamento

8085		8086/8088	
Imediato	MVI A,15H	Imediato	MOV AX, 1000H
Por registrador	MOV A,B	Por registrador	MOV AX,BX
Direto	JMP 2005H	Absoluto ou direto	MOV AX,[1000H]
Indireto por registrador	MOV M,A	Indireto por registrador	MOV AX,[BX]
		Indexado	MOV AX,0100H[BX]
		Baseado	MOV [BX + 0100H], AX
		Baseado e indexado	MOV AX, [BX+SI]
		Baseado e indexado com deslocamento	MOV AX, [BX+SI+5]
		Strings	MOVSB

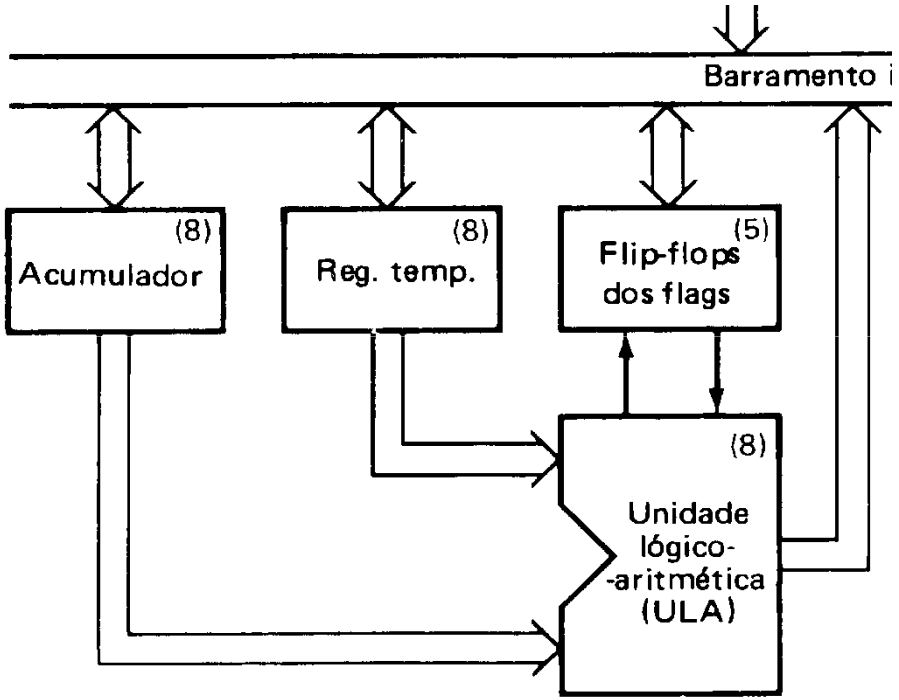
Grupos de Instruções

8085		8086/8088	
Transferência de dados	MOV A,B MVI A,15H	Transferência de dados	MOV AX,BX MOV DL,23H
Aritmético	ADD B SUB B	Aritmético	ADD SI,DX SUB AX,DX
Lógico	ANA B ORI 0FH	Lógico	NOT BX AND CX,DX
Desvio	JMP 2005H JNZ 2010H	Desvio	JMP BX
Controle, Pilha, E/S	PUSH PSW	Controle	CLC STC
		Strings	MOVSB STOSW

Registrador de Flags
8085 - 8088/8086 - 8051

Flags no 8085

Registrador F: Registra o estado da última operação realizada na ULA



Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
S	Z	×	AC	×	P	×	CY

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
S	Z	×	AC	×	P	×	CY

Flag de Sinal: Assume valor 0 para número positivo (bit 7 = 0) e 1 para negativo (bit 7 = 1)

Flag de Zero: Assume valor 0 para número diferente de zero e 1 para número igual a zero.

Flag Auxiliar de Carry: Assume valor 1 quando há transporte do Bit 3 para o Bit 4

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
S	Z	×	AC	×	P	×	CY

Flag de Paridade: Assume valor 1 quando há uma quantidade par de dígitos 1no acumulador. Assume valor 0 quando há uma quantidade ímpar.

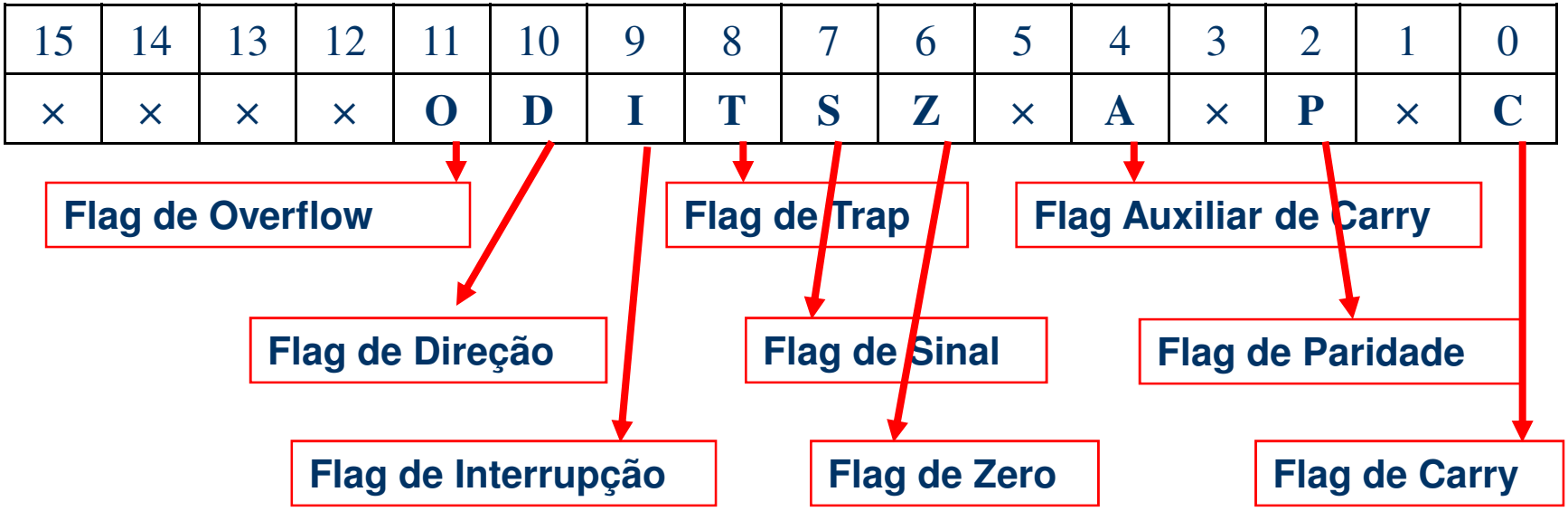
Flag de Carry: Assume valor 1 quando há transporte do Bit 7 para o bit 8 (O Bit 8 é fora do acumulador)

Registrador de Flags – 8086
(Registrador de Estado do Programa (PSW)

É um registrador de 16 bits, mas apenas 9 bits são usados para as flags:

Seis deles são bits de status que refletem os resultados de operações aritméticas e lógicas.

Três são bits de controle.



Registrador de Flags – 8086
(Registrador de Estado do Programa (PSW)

Flags de Status

C – Flag de carry – reflete o ´vai um´ do bit mais significativo, nas operações aritméticas (de 8 ou 16 bits). Ele também é modificado por algumas instruções de rotação e deslocamento. Nas operações de subtração (aritmética em complemento dois) o ´carry´ é invertido e passa a funcionar como ´borrow´ (empréstimo). Se, após uma operação de subtração, obtém-se C = 1, isso indica que não houve ´borrow´, mas C=0, indica que houve ´borrow´.

P – Flag de Paridade – indica a paridade (par), dos 8 bits menos significativos, do resultado da operação realizada.

P = 1 → número par de ´1´ nos 8 bits menos significativos
P = 0 → número ímpar de ´1´ nos 8 bits menos significativos

A – Flag Auxiliar de Carry – reflete o ´vai um´ do bit 3, em uma operação de 8 bits.

Registrador de Flags – 8086
(Registrador de Estado do Programa (PSW)

Flags de Status

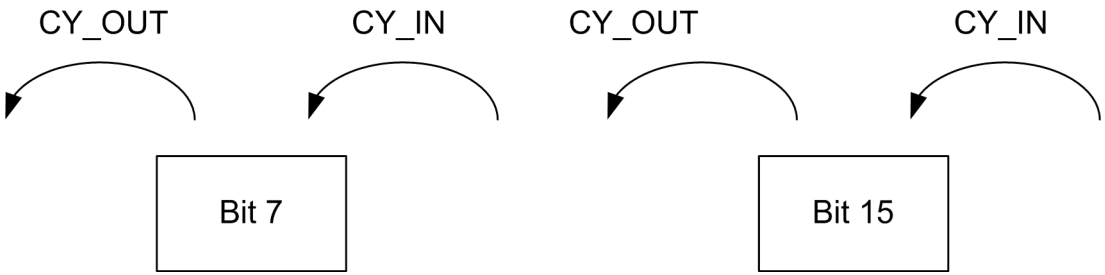
Z – Flag de Zero – indica se uma operação teve zero como resultado.
Z = 1 → se o resultado da operação for igual a zero
Z = 0 → se o resultado da operação for diferente de zero

S – Flag de Sinal – é igual ao bit de mais alta ordem do resultado de uma operação aritmética.
S = 0 → resultado positivo
S = 1 → resultado negativo

O – Flag de Overflow – seu conteúdo é obtido através de uma operação XOR do ´carry in´ com o ´carry out´ do bit de mais alta ordem do resultado de uma operação aritmética. Ele indica um ´overflow´ de magnitude, em aritmética binária com sinal. Indica que o resultado é muito grande para o campo destino.

Registrador de Flags – 8086
(Registrador de Estado do Programa (PSW)

O – Flag de Overflow – seu conteúdo é obtido através de uma operação XOR do ´carry in´ com o ´carry out´ do bit de mais alta ordem do resultado de uma operação aritmética.



CY_IN	CY_OUT	OV = (XOR)
0	0	0
0	1	1
1	0	1
1	1	0

Registrador de Flags – 8086
(Registrador de Estado do Programa (PSW)

Flags de Controle

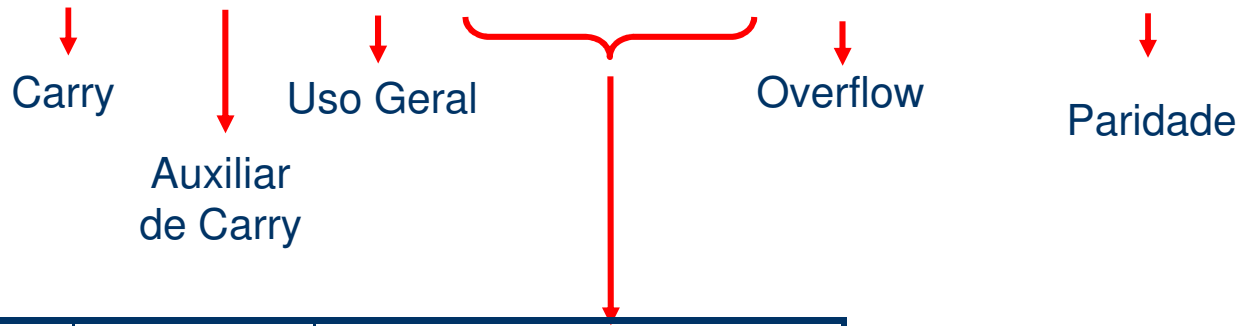
T – Flag de Trap (armadilha) – usada para a depuração de programas. Coloca o 8086 no modo passo a passo. Após cada instrução uma interrupção é gerada automaticamente.

I – Flag de Interrupção – habilita ou desabilita a interrupção externa (pedida pelo pino INTR). Ao contrário do 8085, onde as interrupções RST 7.5, RST 6.5 e RST 5.5 podem ser habilitadas/desabilitadas individualmente, no 8086 todas são habilitadas ou desabilitadas ao mesmo tempo. A habilitação/ desabilitação individual pode ser feita através do controlador de interrupção 8259.
I = 1 → interrupção habilitada I = 0 → interrupção desabilitada

D – Flag de Direção – determina se as operações com ´*strings*´ vão incrementar ou decrementar os registradores de indexação (SI e DI).
D = 1 → SI e DI serão decrementados, ou seja, a ´string´ será acessada a partir do endereço mais alto em direção ao mais baixo.
D = 0 → SI e DI serão incrementados, ou seja, a ´string´será acessada a partir do endereço mais baixo em direção ao mais alto.

Registrador de Flags – 8051

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CY	AC	F0	RS1	RS0	OV	×	P



RS1	RS0	Banco Selecionado
0	0	0
0	1	1
1	0	2
1	1	3

Registrador de Flags – 8051

OV → overflow → A flag de overflow é setada quando há um carry do bit 7, mas não do bit 6 ou um carry do bit 6, mas não bit 7.

A flag de overflow é útil em operações com número sinalizado representados na forma de complemento de 2. Há duas situações que resultam em OV setado:

• Se a soma de dois números positivos for maior que 7F H e menor que FFH a flag de overflow indica que o número não deve ser interpretado como número negativo.

• Se a soma de dois números negativos (bit 7 = 1) resultar em um número no intervalo de 00 H a 7F H (ou 100 H a 17F H, considerando a flag de carry, que sempre estará presente nessa situação), a flag de overflow indicará que o número não é pra ser interpretado como número positivo.

Registrador de Flags – 8051

Exemplos para a flag de overflow:

1. MOV A,#100 → 64H = 0 1 1 0:0 1 0 0
   ADD A,#44 → 2CH = 0 0 1 0:1 1 0 0

Resultado: 144 = 90 H = 1 0 0 1:0 0 0 0 (**OV = 1**)

Decimal		Hexadecimal		1	1		1	1			
100		64H		0	1	1	0	0	1	0	0
44		2CH		0	0	1	0	1	1	0	0
144		90H		1	0	0	1	0	0	0	0

Há transporte do bit 6 para o 7, mas não há do bit 7 para o bit 8.
Os dois números (64H e 2CH) são positivos na operação com sinal. Assim, o resultado deve ser interpretado como número positivo, mesmo tendo bit 7 igual a 1.

Registrador de Flags – 8051

Exemplos para a flag de overflow:

2. ADD,#01H (Ao resultado da operação anterior: 90H)

Resultado: 145 = 91 H = 1 0 0 1:0 0 0 1 (**OV = 0**)

144		90H		1	0	0	1	0	0	0	0
01		01H		0	0	0	0	0	0	0	1
145		91H		1	0	0	1	0	0	0	1

Não há transporte do bit 6 para o 7, nem do bit 7 para o bit 8.
O número 90H é negativo nas operações com sinal e o número 01H é positivo.
Nesse caso não há flag de overflow.

Pilha no 8085 e 8051

Região de memória usada para guardar endereço de retorno e valores temporários

Instruções que usam a pilha:

CALL

RET

PUSH

POP

Interrupções

Endereço	Mnemônico	Código
2000 H	LXI SP,20FFH	**31** FF 20
2003 H		
2004 H		
	XXXXX	XXXXX
	XXXXX	XXXXX
20FF H		

SP →

Transferência de dados envolve sempre um par de bytes → 16 bits

⇩

O byte mais significativo é sempre guardado primeiro na pilha

O byte menos significativo é retirado primeiro da pilha

O Ponteiro de Pilha SP aponta sempre para o topo da pilha (último valor armazenado)

Registradores utilizados → PSW ➔ A + Flags

B ➔ B + C

D ➔ D + E

H ➔ H + L

PUSH reg16 → guarda conteúdo do registrador de 16 bits na pilha

1. O valor de SP é decrementado em 1
2. O byte mais significativo é armazenado na posição SP – 1
3. O valor de SP é decrementado em 1
4. O byte menos significativo é armazenado na posição SP – 2

POP reg16 → carrega registrador de 16 bits com conteúdo da pilha

1. O conteúdo apontado por SP é copiado para o byte menos significativo
2. O valor de SP é incrementado em 1
3. O conteúdo apontado por SP + 1 é copiado para o byte mais significativo
4. O valor de SP é incrementado em 1

Exemplo: A = 01 H, F = 23 H, B = 45 H, C = 67 H

Flags

PUSH PSW

Endereço da RAM	Conteúdo	
2089		
208A		
208B		
208C		
208D		
208E	23	← F
208F	01	← A
2090		
SP após a instrução: 208Eh		

SP inicial: 2090h

PUSH B

Endereço da RAM	Conteúdo	
2089		
208A		
208B		
208C	67	← C
208D	45	← B
208E	23	← F
208F	01	← A
2090		
SP após a instrução: 208Ch		

Exemplo: A = 01 H, F = 23 H, B = 45 H, C = 67 H

POP B

	Endereço da RAM	Conteúdo
	2089	
	208A	
	208B	
	208C	67
	208D	45
F →	208E	23
A →	208F	01
	2090	
	SP após a instrução: 208Eh	

POP PSW

Endereço da RAM	Conteúdo	
2089		
208A		
208B		
208C	67	← C
208D	45	← B
208E	23	← F
208F	01	← A
2090		
SP após a instrução: 2090h		

Pilha

Exemplo de aplicação

Endereço	Mnemônico	Código
2000 H	LXI SP,203FH	**31** 22 20
2003 H	MVI A,20H	**3E** 20
2005 H	PUSH PSW	**F5**
2006 H	CALL MOSTRAA	**CD** 6E 03
2009 H	POP PSW	**F1**
200A H	ADI 01H	**C6** 01
200C H	**JNZ** 2005 H	**DA** 05 20
200F H	JMP 2003	**C3** 03 20
2022 H		

Pilha no 8051

Transferência de dados envolve apenas um byte → 8bits

Registradores de 16 bits, como DPTR, são guardados em duas operações PUSH e retiradas em duas operações POP

Exemplos:
PUSH ACC → Guarda conteúdo do acumulador
PUSH DPH → Guarda 8 bits mais significativos do DPTR
PUSH DPL → Guarda 8 bits menos significativos do DPTR
POP ACC → Recupera conteúdo do acumulador

Ao contrário do 8085, o apontador de pilha SP é incrementado no 8051, nas operações PUSH.

PUSH reg8 → guarda conteúdo do registrador de 8bits na pilha

1. O valor de SP é incrementado em 1
2. O byte é armazenado na posição SP + 1

POP reg18→ carrega registrador de 8bits com conteúdo da pilha

1. O conteúdo apontado por SP é copiado para o registrador indicado
2. O valor de SP é decrementado em 1

Diferenças básicas entre o uso da Pilha no 8051 e no 8085

Característica	8085	8051
Variação do apontador de pilha SP	SP é decrementado nas operações para guardar valores na pilha (PUSH, CALL, chamada de interrupção)	SP é incrementado nas operações para guardar valores na pilha (PUSH, CALL, chamada de interrupção)
Região da memória	A região da pilha é a mesma do programa do usuário, podendo haver sobreposição	A pilha é reservada na memória RAM e o programa na memória ROM, não havendo risco de sobreposição
Tamanho do dado guardado	São movimentados 16 bits em cada operação de pilha	São movimentados 8 bits em cada operação de pilha

Bibliografia

[1] ZILLER, Roberto M., "Microprocessadores – Conceitos Importantes," Edição do autor, Florianópolis, 2000. ISBN 85-901037-2-2

[2] MALVINO, Albert Paul, "Microcomputadores e microprocessadores; tradução Anatólio Laschuk, revisão técnica Rodrigo Araês Farias. São Paulo: McGraw-Hill do Brasil, 1985.