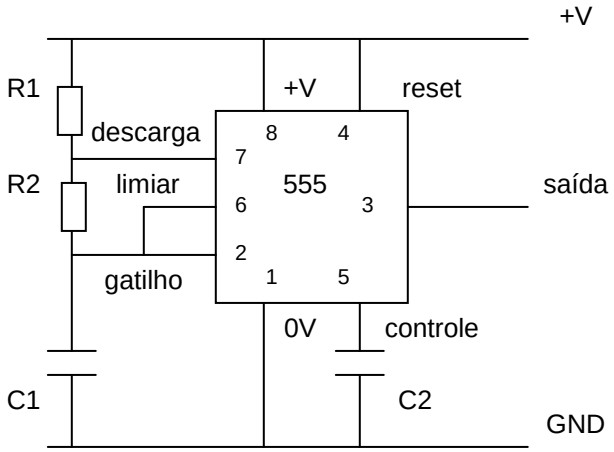
Controle de temporização de circuitos

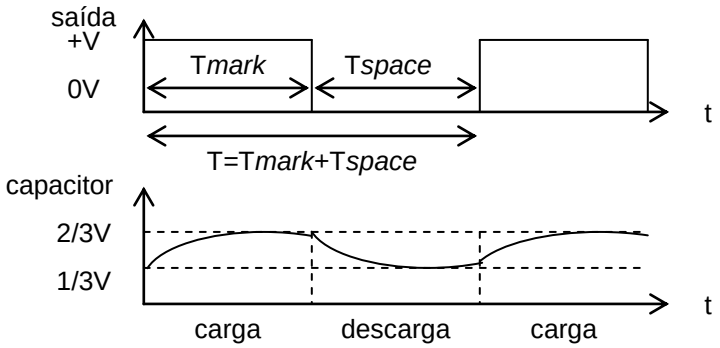
Um dos circuitos integrados mais versáteis para controle de tempo é o 555, capaz de funcionar em três modos:

- monoestável — modo no qual o circuito produz um único disparo
  aplicações: temporização, chaveamento sem ressaltos, divisores de frequência, modulação de largura de pulso (PWM) etc.
- astável — modo no qual o circuito opera como um oscilador,
  capaz de alternar regularmente entre estados altos e baixos
  aplicações: acionamentos de LEDs, geradores de tons, alarmes, modulação de posição de pulso, ***clocks*** etc.
- biestável — modo no qual o circuito opera como um ***flip-flop***,
  capaz de permanecer em um de dois estados indefinidamente
  aplicações: chaveamento sem ressaltos (***bouncefree latched***) e registradores (memória)

O diagrama abaixo representa a configuração típica para um oscilador em modo astável:



O capacitor C1 é carregado pela corrente que passa por R1 e R2. Quando a carga alcança 2/3 da tensão de alimentação (+V), o limiar é atingido, a saída vai para nível baixo e o pino de descarga é conectado a 0V. Quando a descarga da corrente que passa por R2 atinge 1/3 da tensão de alimentação, a saída vai para nível alto e cessa a descarga permitindo a recarga do capacitor. O ciclo se repetirá continuamente até que o pino de ***reset*** seja conectado a 0V.



Um ciclo de trabalho (carga e recarga) ocorre durante o período (T) da onda quadrada, o qual inclui o tempo de marcação (Tm) e o tempo de espaçamento (Ts):

$$T = Tm + Ts = [0,7 \times (R1 + R2) \times C1] + [0,7 \times R2 \times C1] = 0,7 \times (R1 + 2R2)$$

onde

T – período [s]
Tm – tempo de marcação [s]
Ts – tempo de espaçamento [s]
R1 – resistor [ohms]
R2 – resistor [ohms]
C1 – capacitor [F]

A freqüência de oscilação [Hz] é o número de ciclos de trabalho por segundo:

$$f = \frac{1}{T} = \frac{1,4}{(R1 + 2R2) \times C1}$$

Para que o circuito funcione no modo astável, o tempo de marcação (Tm) deverá ser praticamente igual ao tempo de espaçamento (Ts). Isso acontecerá se o valor de R2 for muito maior que R1, nesse caso o valor da frequência será dado por:

$$f = \frac{0,7}{R2 \times C1}$$

Exemplo:

Com os valores dados abaixo:

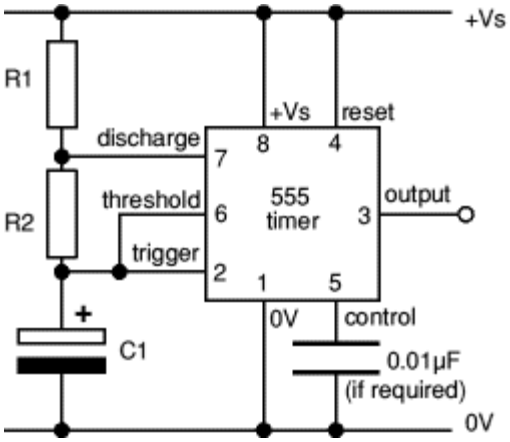
R1 = 1 KΩ
R2 = 68 KΩ
C1 = 10 μF
C2 = 0,1 μF (para estabilização)

A frequência será de

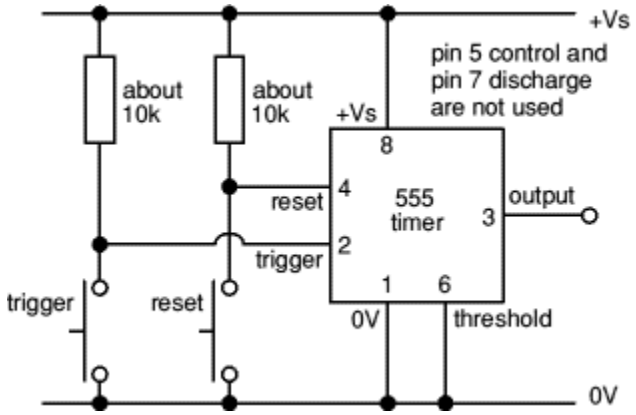
$f = 0,7 / (68 \times 10^{3} \times 10 \times 10^{-6}) \approx 1$ Hz

Exemplos de usos do temporizador

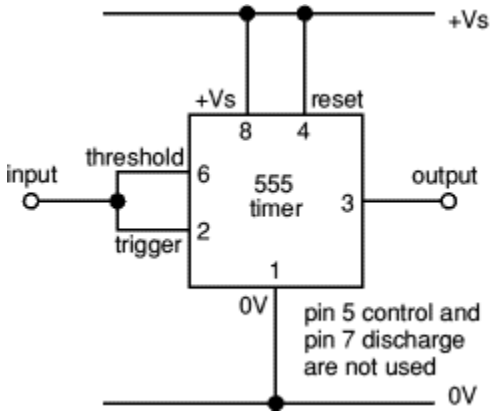
ASTÁVEL



BIESTÁVEL (FLIP-FLOP)

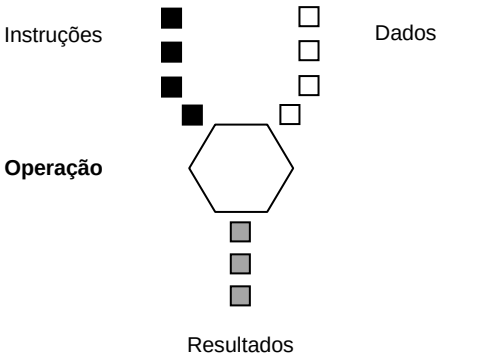


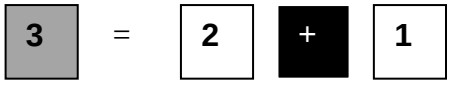
NOT GATE



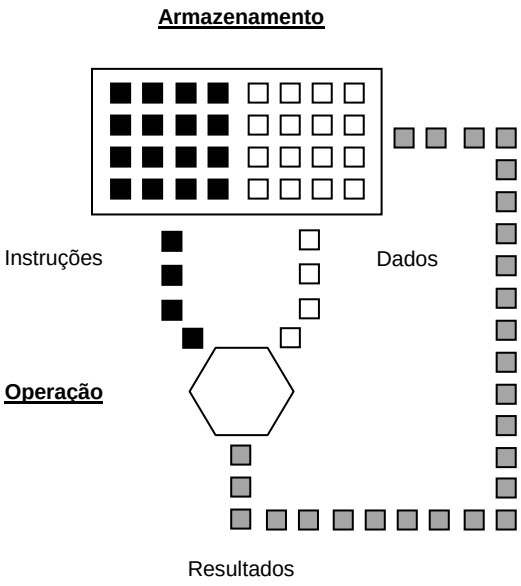
Modelos de computador

Modelo de computador
baseado em operação (1)

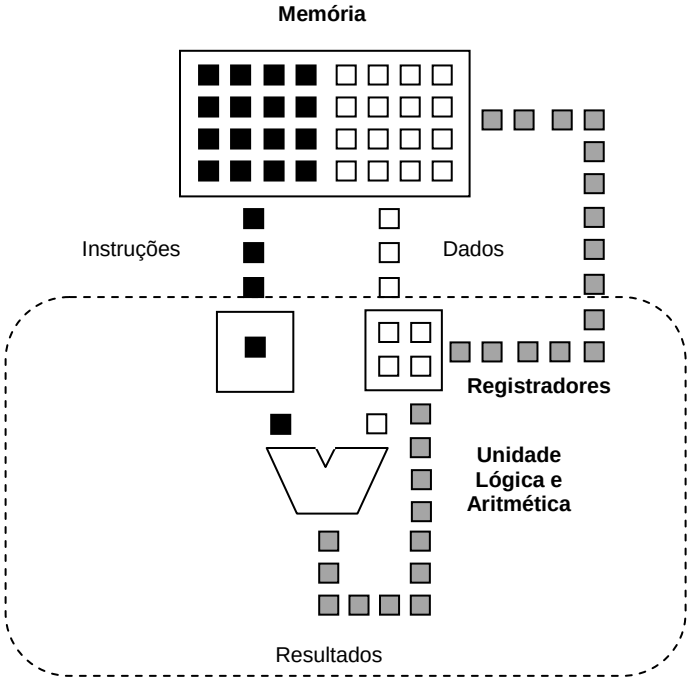




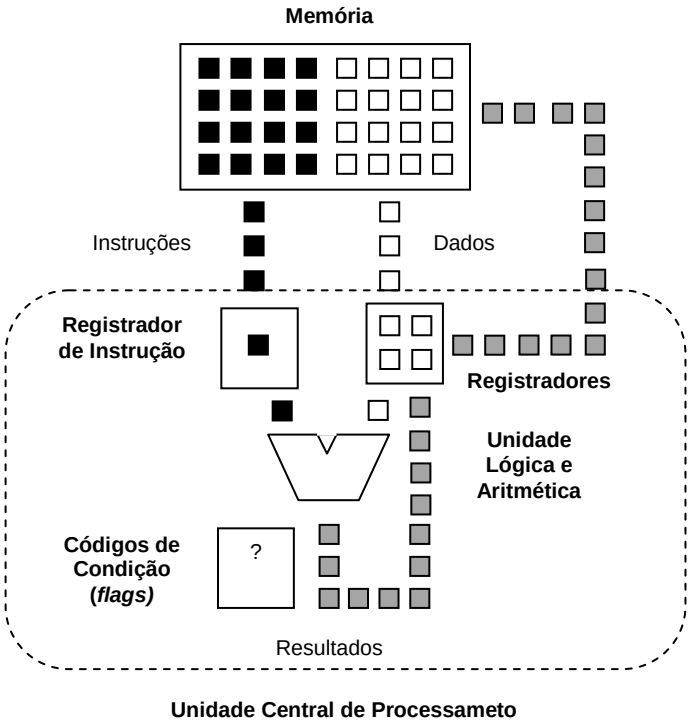
Modelo de computador
com armazenamento de dados e instruções (2)



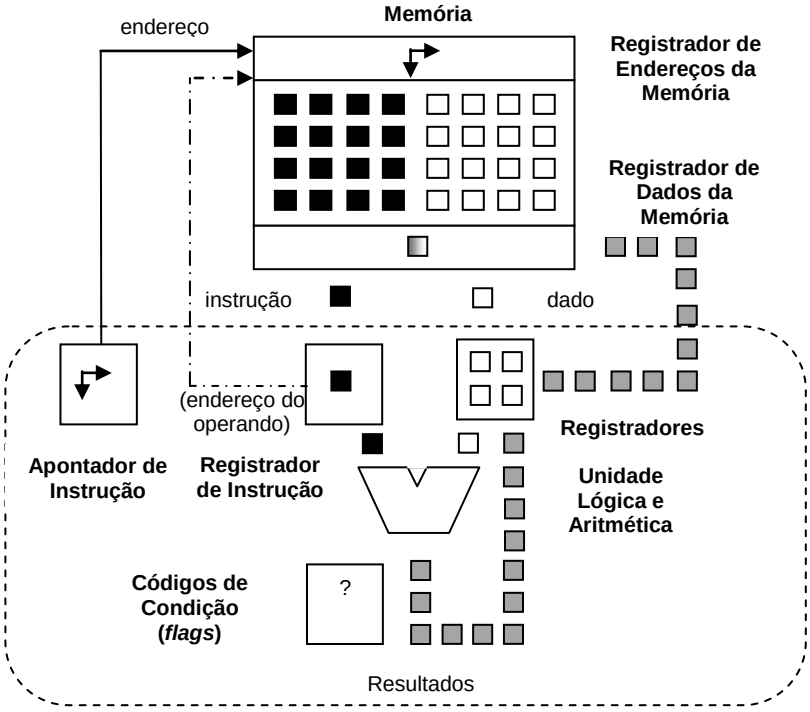
Modelo de computador
com banco de registradores para dados (3)



Modelo de computador
com registradores para instrução e códigos de condição (4)

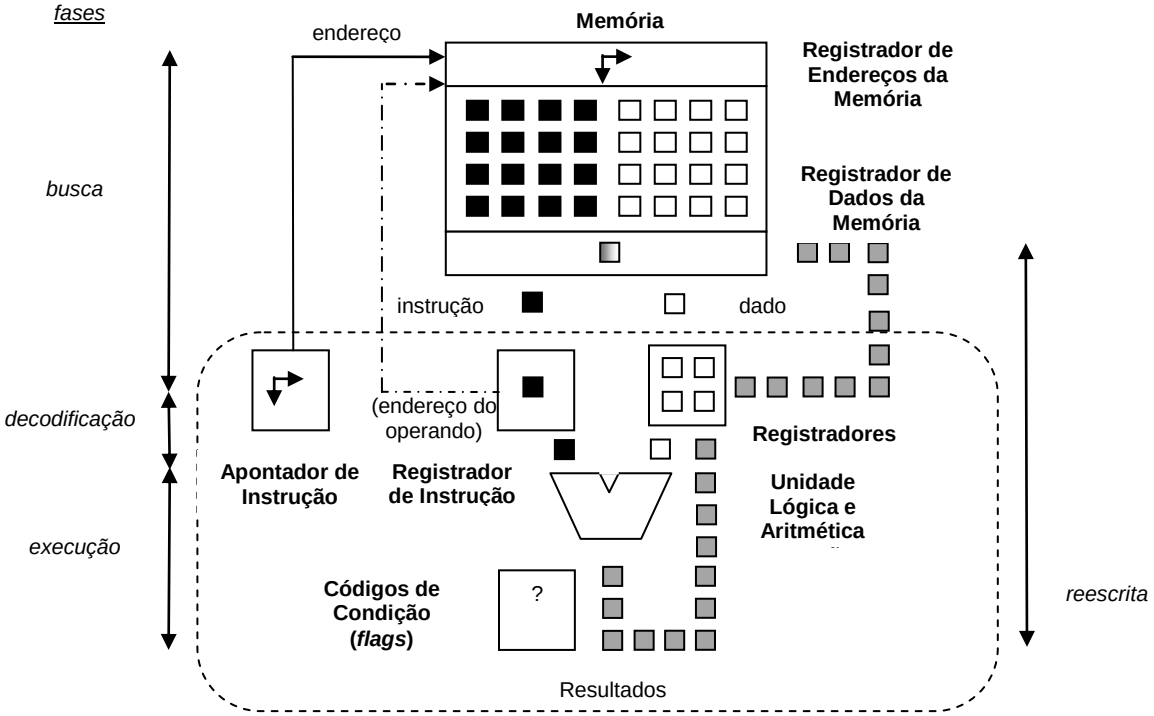


Modelo de computador
com apontador de instrução e memória (5)



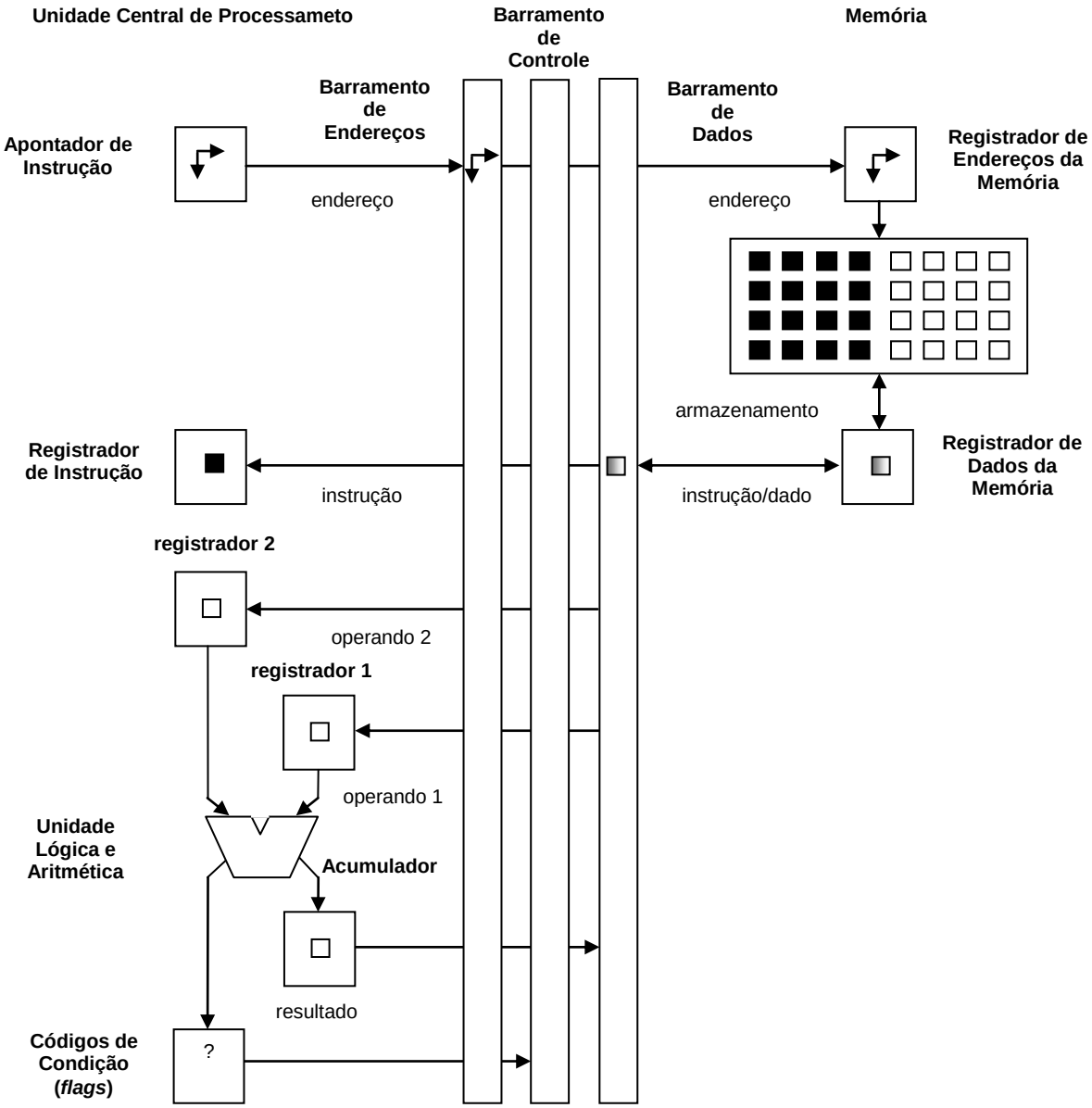
Unidade Central de Processameto

Modelo de computador
e fases de funcionamento (6)



Unidade Central de Processameto

Modelo de computador
com barramentos e registradores operacionais (7)



Busca de instrução

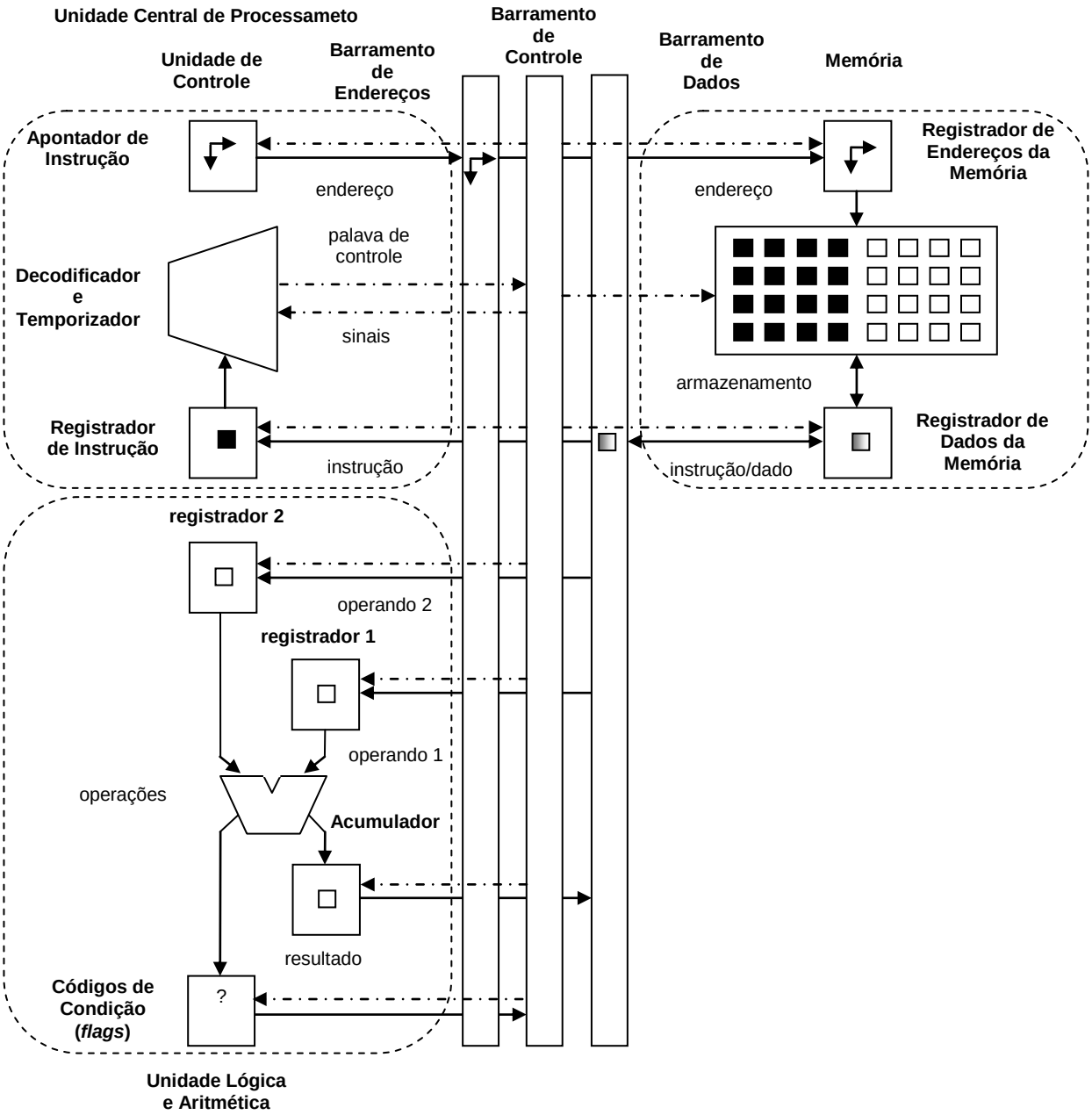
R.E.M. ← A.I. (endereço da instrução)
R.D.M ← MEM [R.E.M.]
R.I. ← R.D.M. (cópia da instrução)
A.I. ← A.I. + 1 (endereço da próxima instrução)

Decodificação (identificar operando(s) e operação)

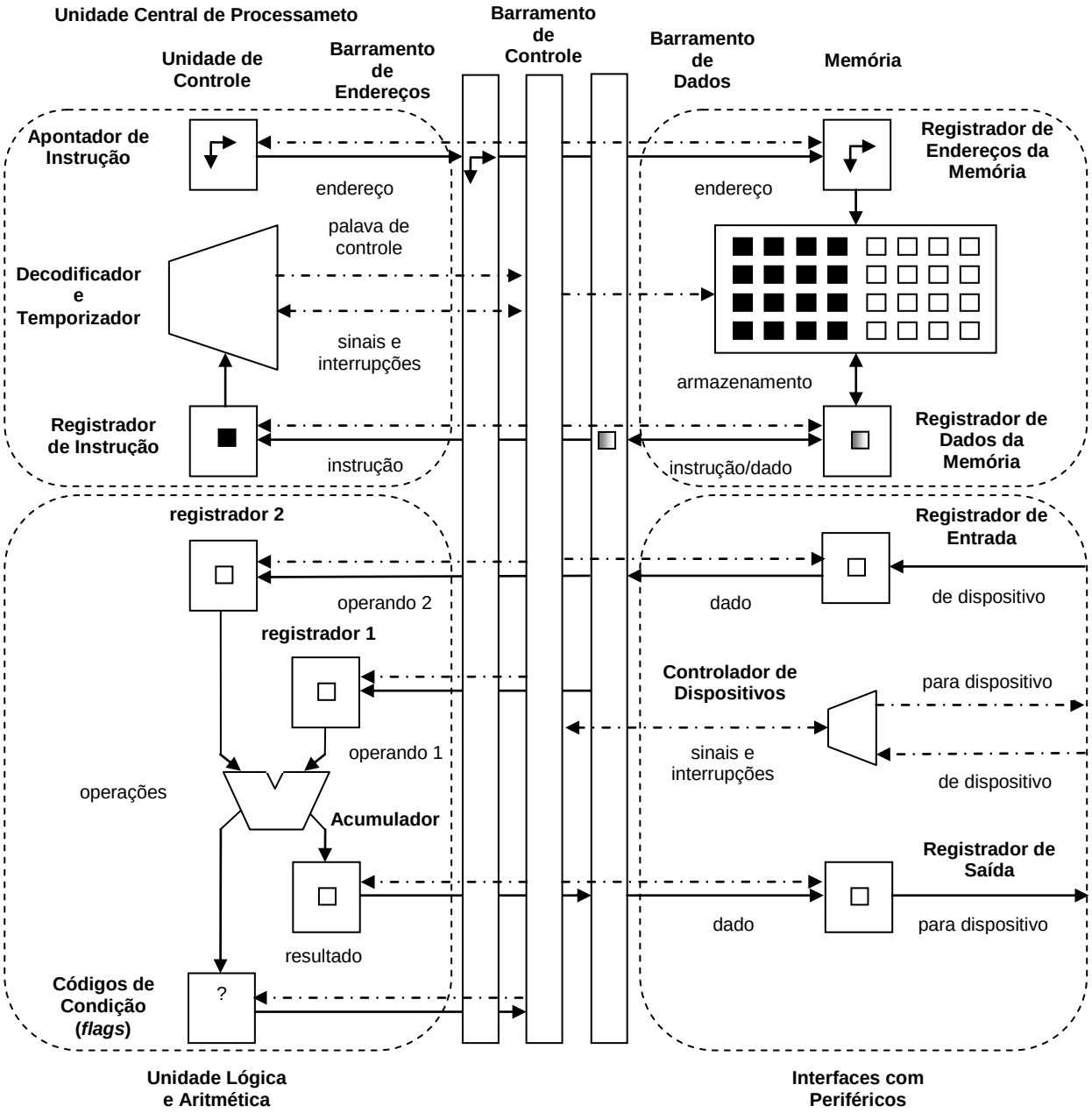
Execução de instrução

R1 ← AC
R.E.M. ← R.I.(endereço do operando)
R2 ← MEM [R.E.M.]
AC ← R1 + R2

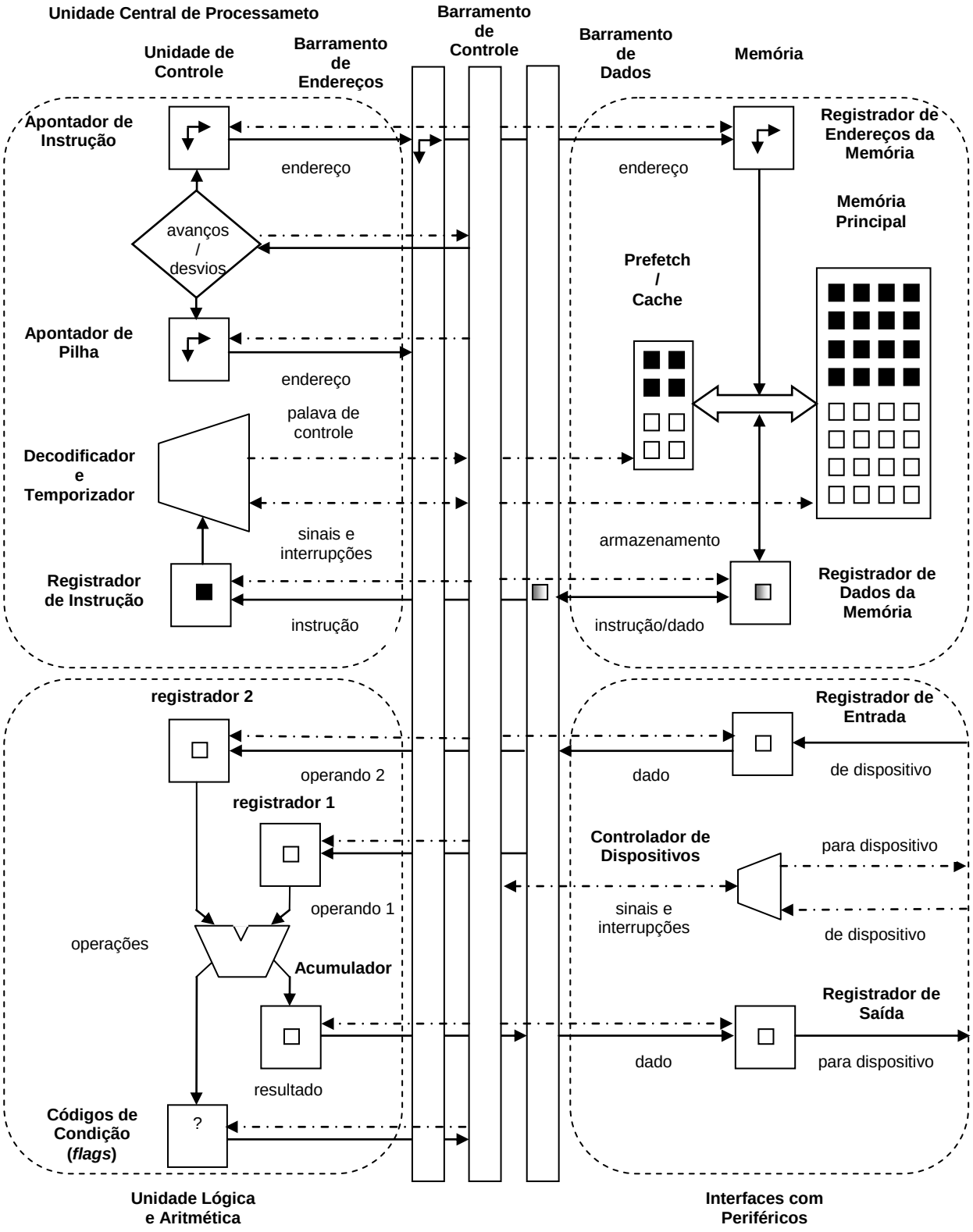
Modelo de computador
com unidades funcionais (8)



Modelo de computador
com controle de periféricos



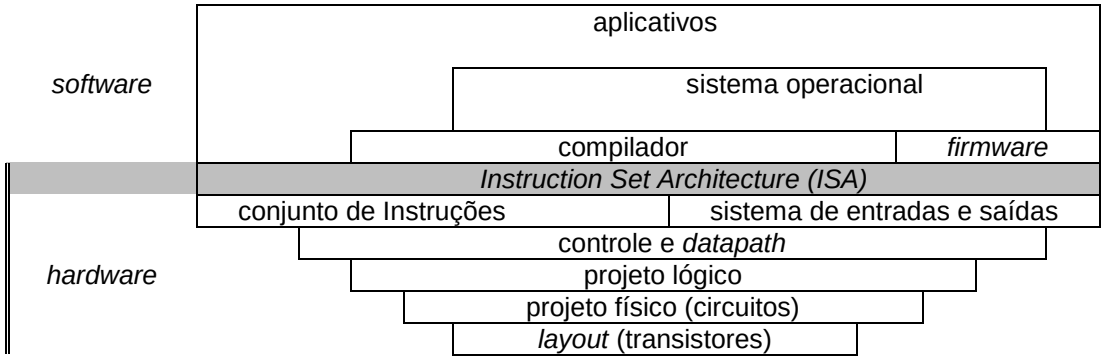
Modelo de computador
com cache e controle de avanços e desvios (10)



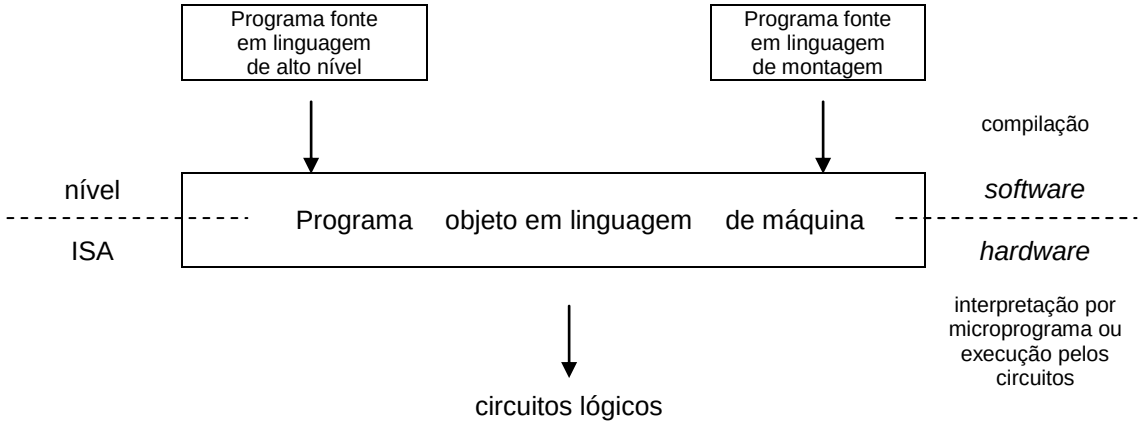
Interface *software-hardware* (*ISA – Instruction Set Architecture*)

O nível ISA define como uma máquina se apresentará ao programador:

- quais as instruções em linguagem de máquina
- qual o modelo de memória (quantidade de bits, alinhamento etc.)
- quantidade e tipos dos registradores (uso geral, apontadores, pilha, de estado)
- tipos de dados disponíveis (numéricos e não-numéricos: lógicos, cadeias etc.)



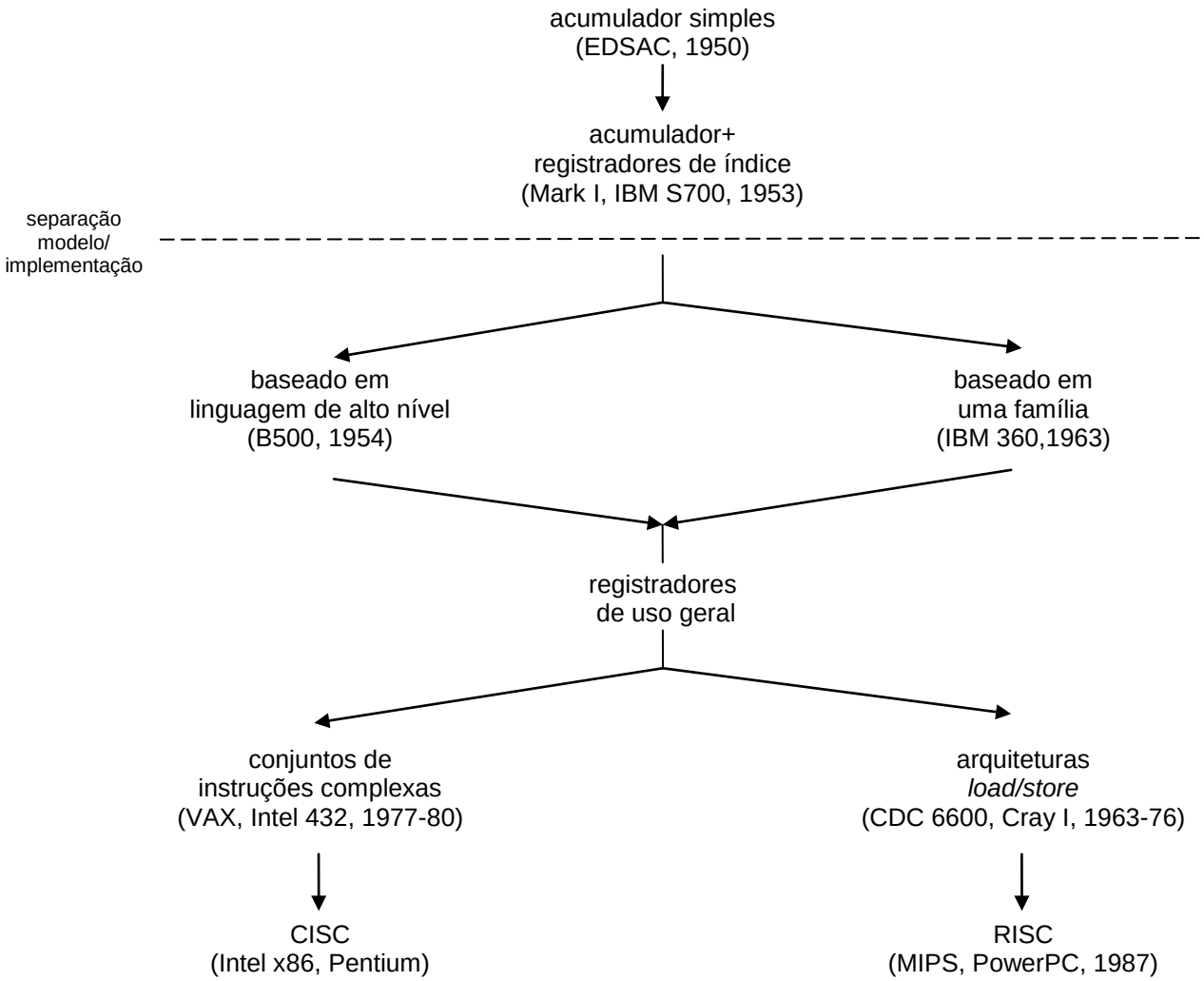
De forma simplificada, o nível ISA estabelece os limites entre *software* e *hardware*.



Em termos práticos, uma instrução será executada por circuitos lógicos, embora cada nível possa ter uma forma diferente para sua expressão:

linguagem de alto nível:	A = 10;
linguagem de montagem:	LDA 0ah
linguagem de máquina:	00110011 00001010

Evolução dos conjuntos de instruções



Classificação de ISA's		C = A + B
	Instrução	Descrição
1. Acumulador		(antes de 1960, 68HC11)
	load A	AC ← mem [A]
1 endereço	add B	AC ← AC + mem [B]
	store C	mem [C] ← AC
2. Pilha		(de 1960 a 1970)
	push A	SP ← SP+1; stack[SP] ← mem [A];
	push B	SP ← SP+1; stack[SP] ← mem [B]
0 endereço	add	stack[SP-1] ← stack[SP-1] + stack[SP]
	pop C	SP ← SP-1 ; mem [C] ← stack[SP];
3. Memória-Memória		(de 1970 a 1980)
2 endereços	add A, B	mem [A] ← mem [A] + mem[B]
3 endereços	add A, B, C	mem [A] ← mem [B] + mem[C]
4. Registrador-Memória		(1970 em diante, 80x86)
	load R1, A	R1 ← mem [A]
2 endereços	add R1, B	R1 ← R1 + mem [B]
	store C, R1	mem [C] ← R1
5. Registrador-Registrador (*load/store*)		(1960 em diante, MIPS)
	load R1, A	R1 ← mem [A]
	load R2, B	R2 ← mem [B]
3 endereços	add R3, R1, R2	R3 ← R1 + R2
	store C , R3	mem [C] ← R3

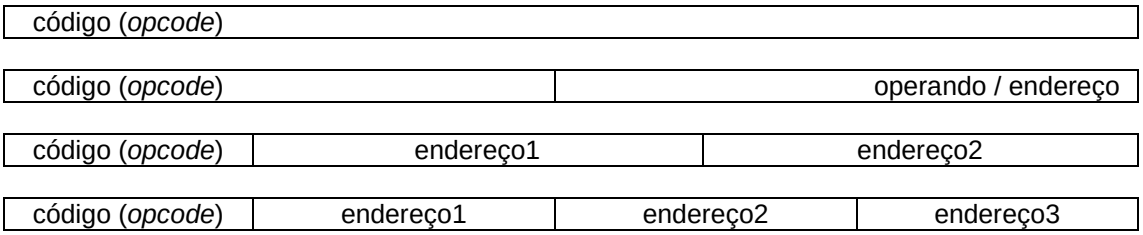
Tipo	Vantagens	Desvantagens
Acumulador	- boa densidade de código - compilador simples de escrever	- acumulador é gargalo - dificulta paralelismo e *pipelining* - compilador otimizado é difícil - alto tráfego com a memória
Pilha	- boa densidade de código - poucos requisitos de hardware - compilador simples de escrever	- pilha é gargalo - dificulta paralelismo e *pipelining* - compilador otimizado é difícil - operações complementares para movimentar dados na pilha
Memória-Memória	- boa densidade de código (3) - compilador simples de escrever	- tempo variável por instrução - operações complementares para lidar poucos operandos - alto tráfego com a memória
Registrador-Memória	- boa densidade de código - possível acessar dado sem carregar	- tempo variável por instrução - baixa ortogonalidade - limitado em registradores
Registrador-Registrador *load/store*	- boa densidade de código - mesmo tempo por instrução - fácil paralelismo e *pipelining*	- instruções numerosas - nem sempre três operandos - dependente de bom compilador
Registradores (1980 em diante)	- mais rápidos do que *cache* - tráfego de memória reduzido	- limitados em quantidade - salvar e restaurar contexto - dependente de bom compilador

Formatos de instruções

O formato das instruções está relacionado ao modelo de memória, à quantidade de processadores, ao tempo de decodificação, ao tempo de execução (busca de operandos) e à quantidade de memória endereçável.

Os formatos podem ser constituídos por códigos de instrução (opcode) e operandos/endereços.

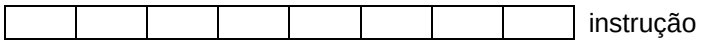
Os formatos podem ser fixos ou variados, com 0, 1, 2 ou mais operandos/endereços, dependendo da arquitetura e dos modos de endereçamento (memória e registradores).



Tipos de formatos de instrução

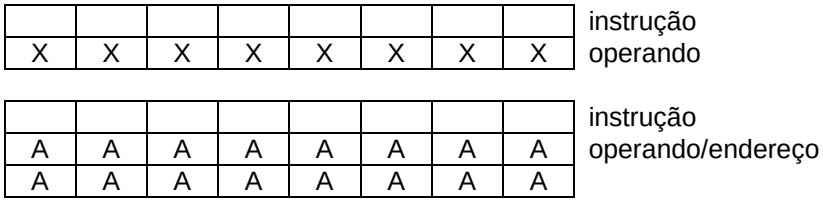
Modos de endereçamento

1. Implícto
   - instrução traz o próprio o endereçamento (não há operando)



Exemplo:
No Intel 8080:
STC – Set Carry Flag
RAL – Rotate Accumulator Left
RLC – Rotate Accumulator through Carry
DAA – Decimal Adjust Accumulator

2. Imediato
   - instrução traz o próprio operando (dado/constante)



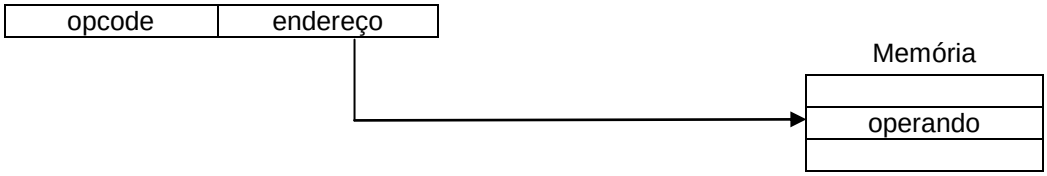
Exemplo:
No Intel 8080:
ADI data – Add Immediate to accumulator
LDI data – Load Immediate to accumulator
JMI address – Jump Immediate to address
CPI data – Compare Immediate with accumulator

Uso:
- valor constante
- valor inicial para contador
- armazenar endereço (ponteiro) em registrador
- indicar quantidade de posições em deslocamento de bits

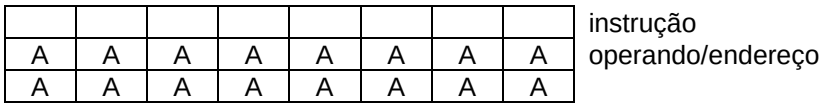
Vantagens:
- uso com valores constantes
- operando obtido durante o ciclo de busca (apenas 1 acesso)

Desvantagens:
- tamanho do dado limitado à quantidade de bits
- não há flexibilidade para se alterar dados que variem a cada execução do programa

3. Direto



- instrução traz o endereço do operando (dado) na memória



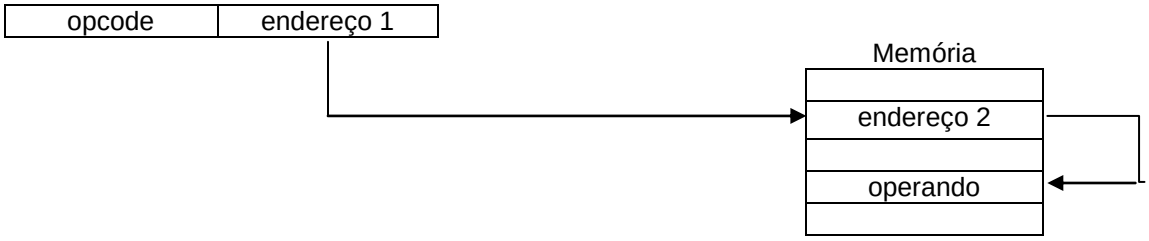
Exemplo:
No Intel 8080:
LDA address – Load accumulator with Address content
JMP address – Jump to address

Uso:
- indicar posição em memória

Vantagens:
- referência direta à memória

Desvantagens:
- tamanho do endereço limitado à quantidade de bits
- mais lento que o modo imediato (mais ciclos para busca do operando durante execução)

4. Indireto



- instrução indica o endereço (1) de outro (2) onde está o dado na memória

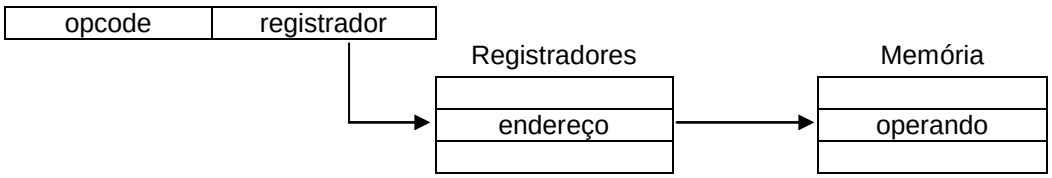
Exemplo:
No Z80:
LD A, (address) – Load Accumulator Indirect from memory

Uso:
- indicação do endereço do dado

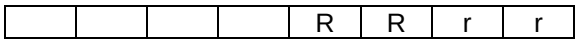
Vantagens:
- referência indireta à memória (estruturas de dados mais complexas)
- com um endereço menor (apontador) indicar dado em um espaço de endereçamento maior

Desvantagens:
- mais lento que o modo imediato (mais ciclos para busca do operando durante execução)

5. Indireto via registrador



- instrução indica o(s) registrador(es) que contém o endereço (apontador) do dado na memória



instrução/registrador(es)

Exemplo:

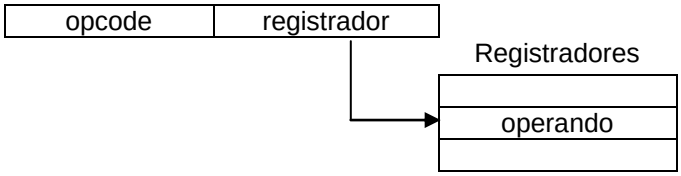
No Intel 8080:
ADD M – Add Memory addressed by register pair (HL) to accumulator
MOV M, register – Move register to Memory addressed by register pair (HL)

Uso:
- indicação do dado

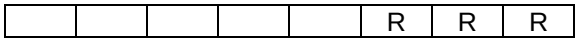
Vantagens:
- referência indireta à memória (estruturas de dados mais complexas)

Desvantagens:
- tamanho de registradores limitado
- mais lento que o modo imediato (mais ciclos para busca do operando durante execução)

6. Direto via registrador



- instrução indica o registrador que contém o dado



instrução/registrador

Exemplo:

No Intel 8080:
ADD register – Add register to accumulator
DCR register – Decrement register

Uso:
- contador

Vantagens:
- tamanho da instrução pequeno
- referência direta a registrador (não faz acesso à memória)

Desvantagens:
- quantidade de registradores limitada
- mais lento que o modo imediato (mais ciclos para busca do operando durante execução)

7. Indexado
- instrução opera sobre o endereço obtido pela soma do operando a um registrador (índice)

Exemplo:
No Intel 8086:
LDX register, operand – Load register with memory addressed by (register+operand)
ADX register, operand – Add register with memory addressed by (register+operand)

Uso:
- para acessar dado em arranjo

Vantagens:
- referência rápida para acesso a dados contíguos na memória

Desvantagens:
- tamanho

8. Modo (base+deslocamento)

- instrução opera sobre o endereço obtido pela soma do operando (deslocamento) ao endereço contido em um registrador (base)

Exemplo:
ADD [base+index_register+offset], register – Add register to memory address

Uso:
- para segmentação e para realocação de dados/programas na memória

Vantagens:
- referência rápida para acesso a porções contíguas na memória

Desvantagens:
- tamanho limitado

9. Combinado
- instruções que combinam modos de endereçamento: direto (ou imediato) + indireto (via pilha)