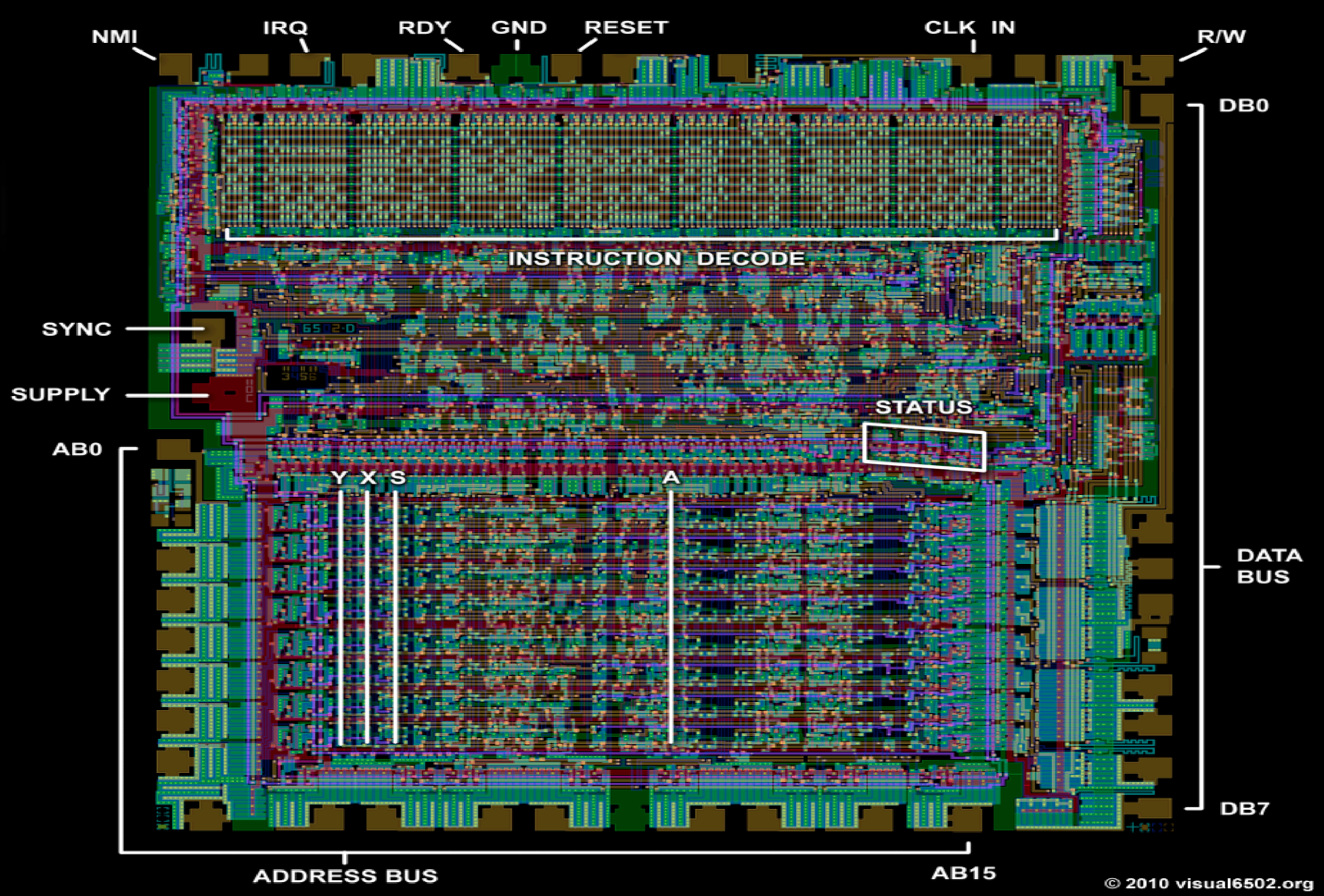
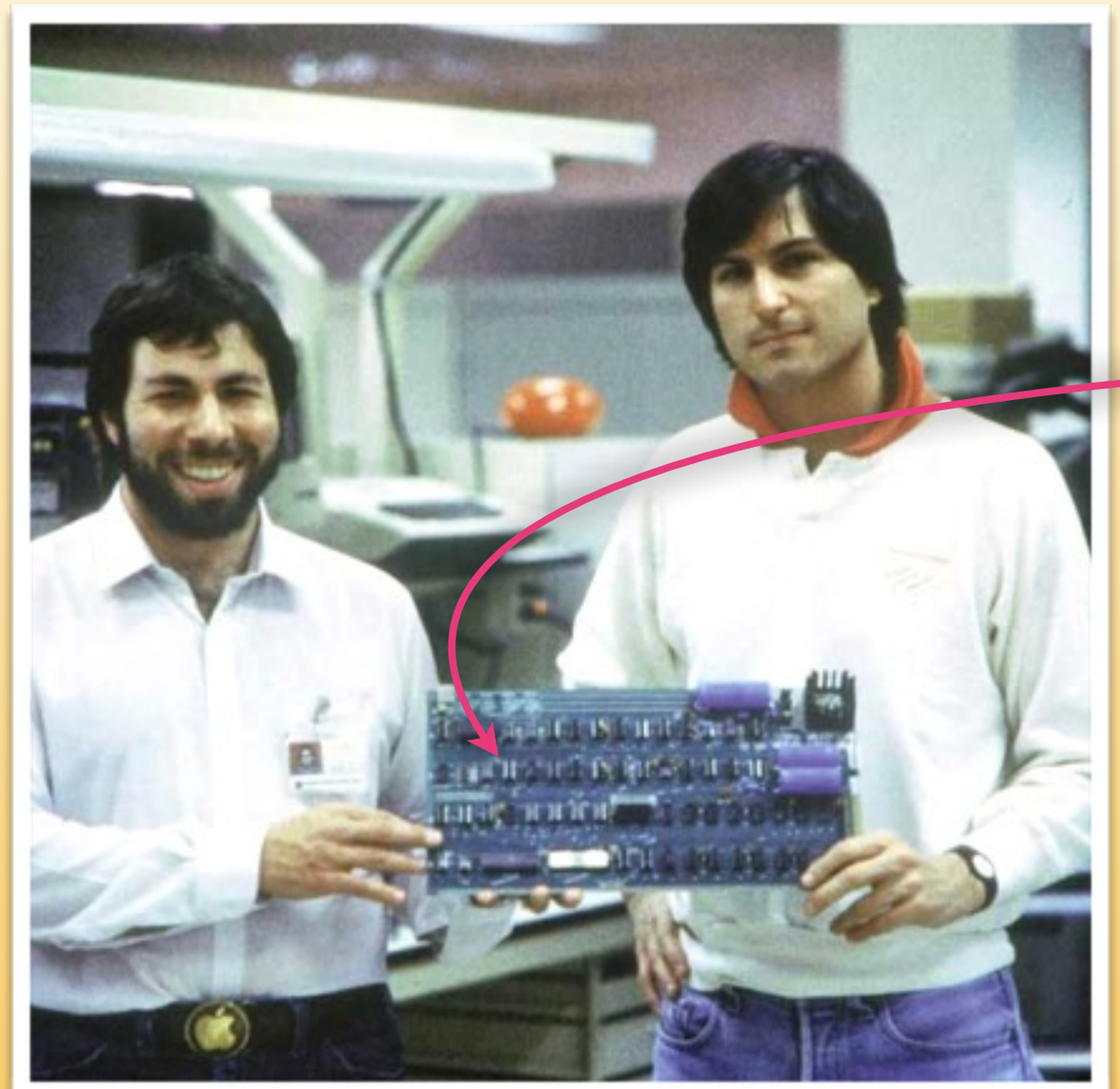


Introdução à Arquitetura de computadores

Fernando Passold ✲ Eng. Elétrica

um pouco de HISTÓRIA



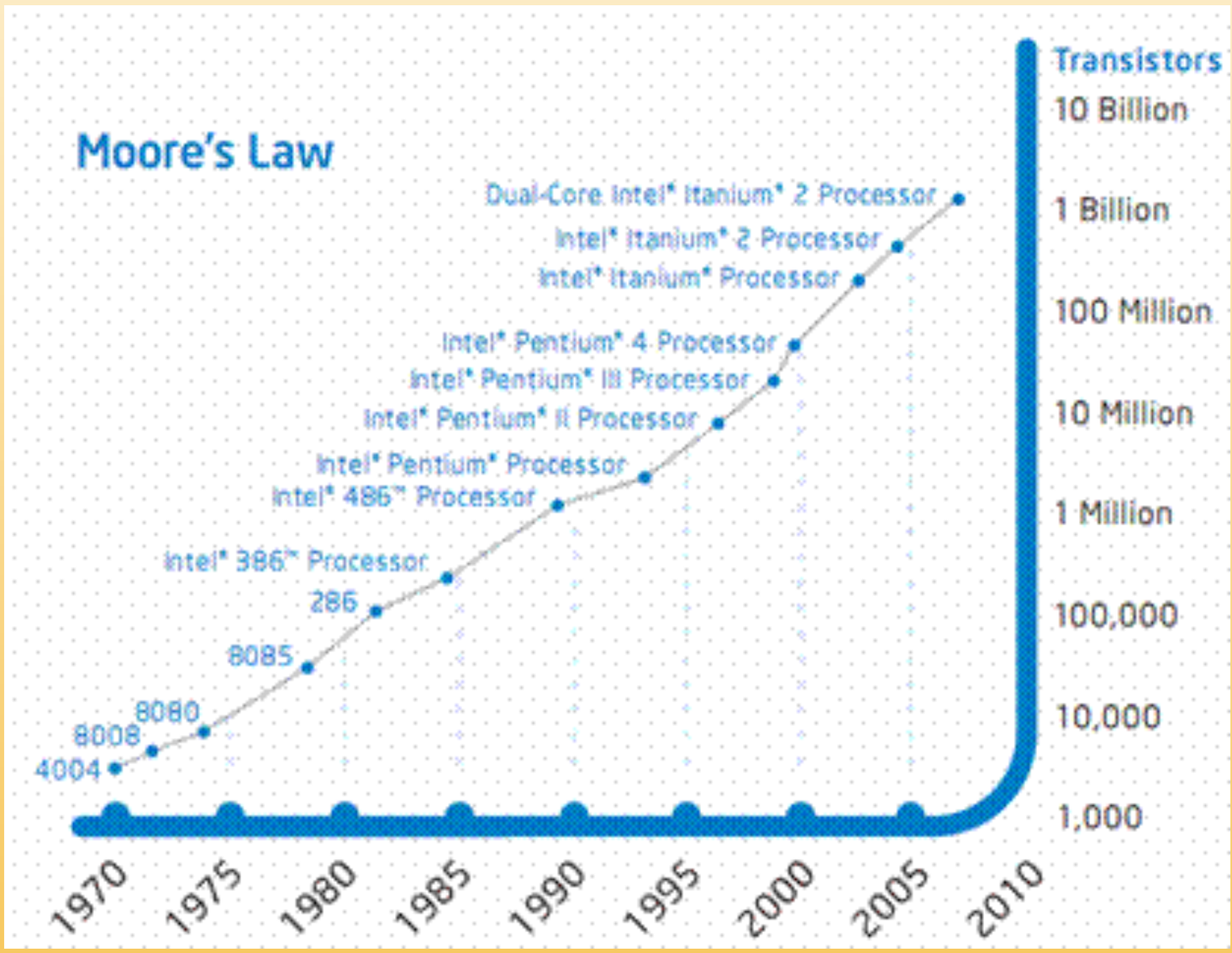
Steve Wozniak, Steve Jobs, and their Apple I (1975).

- 1971: Intel fabrica o 1º microprocessador (**4004**), 4-bits, fabricado a pedido da DataPoint (objetivo: substituir a CPU dos terminais inteligentes).
- 1972: Intel desenvolve o (**8008**), µP de 8-bits, capaz de endereçar 16 Kbytes, mais velocidade.
- 1973: Intel lança o (**8080**), 500.000 operações por segundo, endereça até 64 Kbytes de RAM.
- 1975: Lançamento dos µP 6501 e **6502** (MOS FETs), reduzindo os preços e gerando o CAOS…
- 1976: Zilog lança o **Z80**, µP de 8-bits, mais poderoso, até 64 KBytes RAM, 176 instruções.
- 1976: Surgem os µC's.
- 198x: Ruptura na evolução dos µP x µC. Os µC incorporam capacidades para interagir com o mundo físico em tempo real. Os µP: melhores capacidades para lidar com grandes volumes de dados.

antes de 1980...

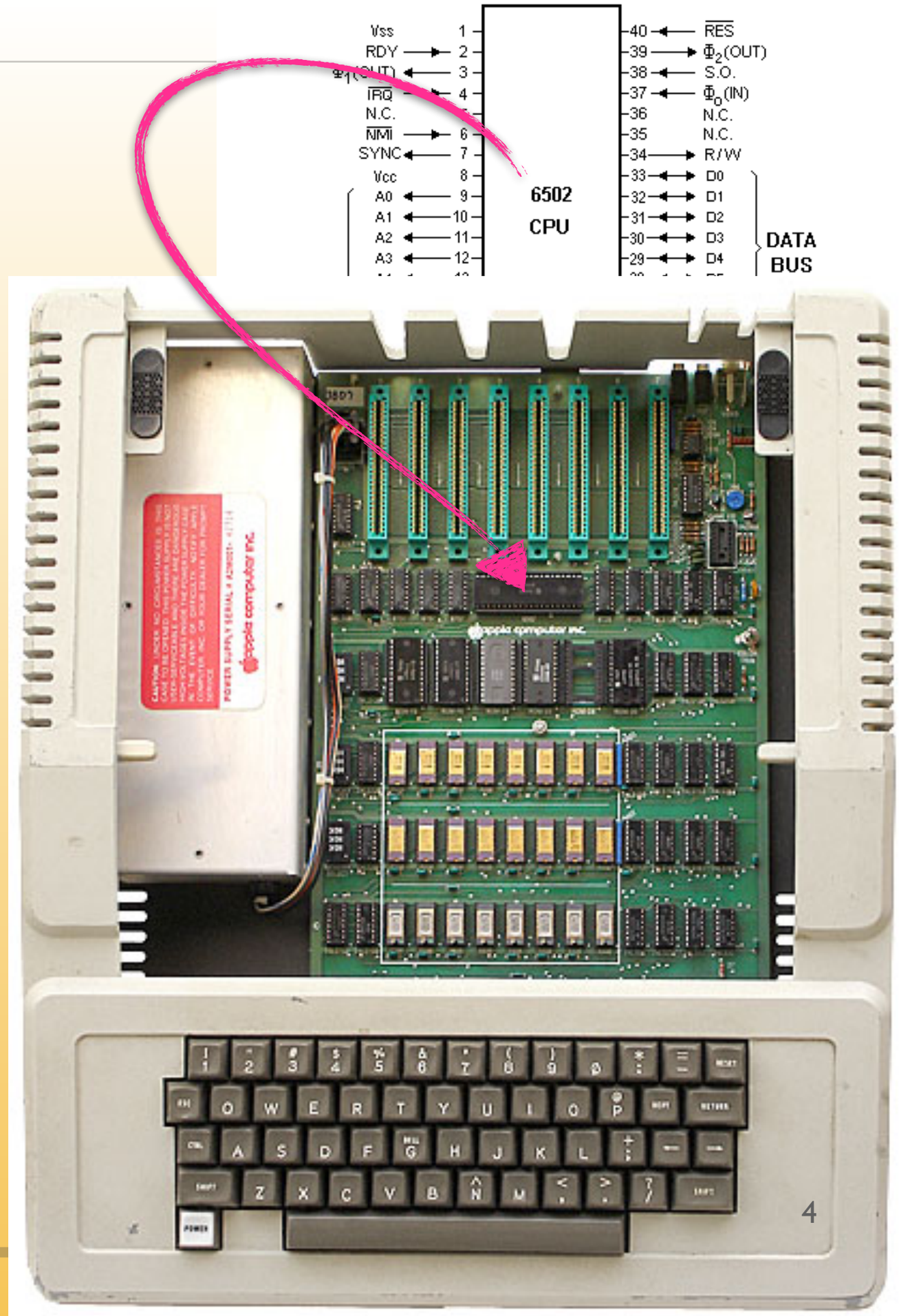
Os chips eram desenhados de forma manual!

Detalhes:



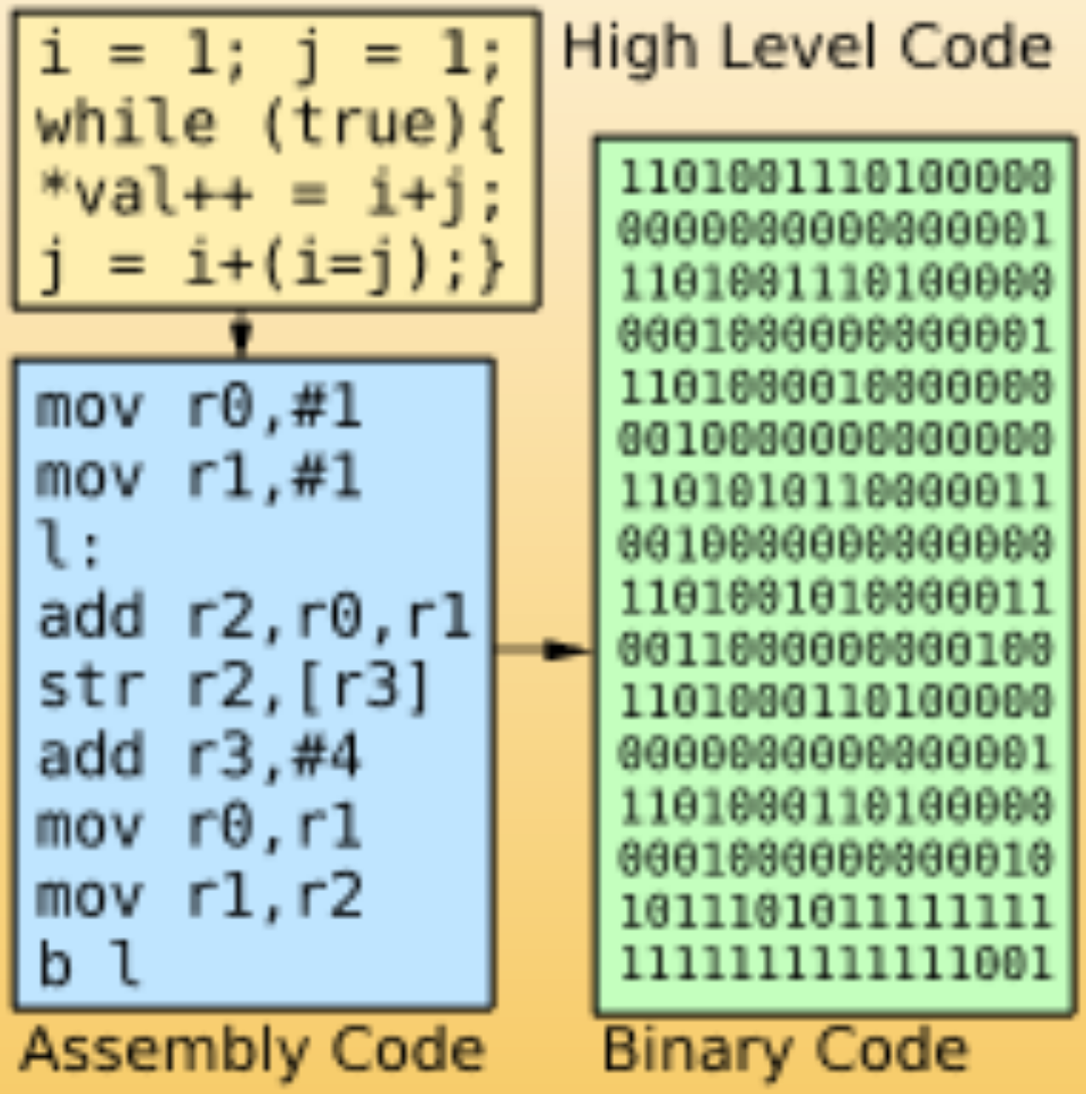
O “clássico" Apple II

- Lançado em 1.977 (40 anos atrás);
- 1981: “upgrade" para Apple II+;
- uP 6502 @ 1,023 MHz:
  8-bits; 40 pinos; 4.529 transístores;
- Um Xeon de 2015 possui 2.500.000.000.000 transístores.
- Capaz de endereçar até 2^16 = 65.536 posições de memória, ou
  64 Kbytes de RAM nativo (barramento de endereços de 16 linhas: A15 ~ A0);
  3 slots para bancos de RAM de 4KB ou 16 KB.
- ROM de 16 KBytes: Apple DOS + Basic;
- Expansível até 1 Mbyte, com cartões extras ➛ Exige: 20 linhas de endereçamento (2^20=1.048.576 / 2^{10} = 1.024 K / 1024 = 1 M).



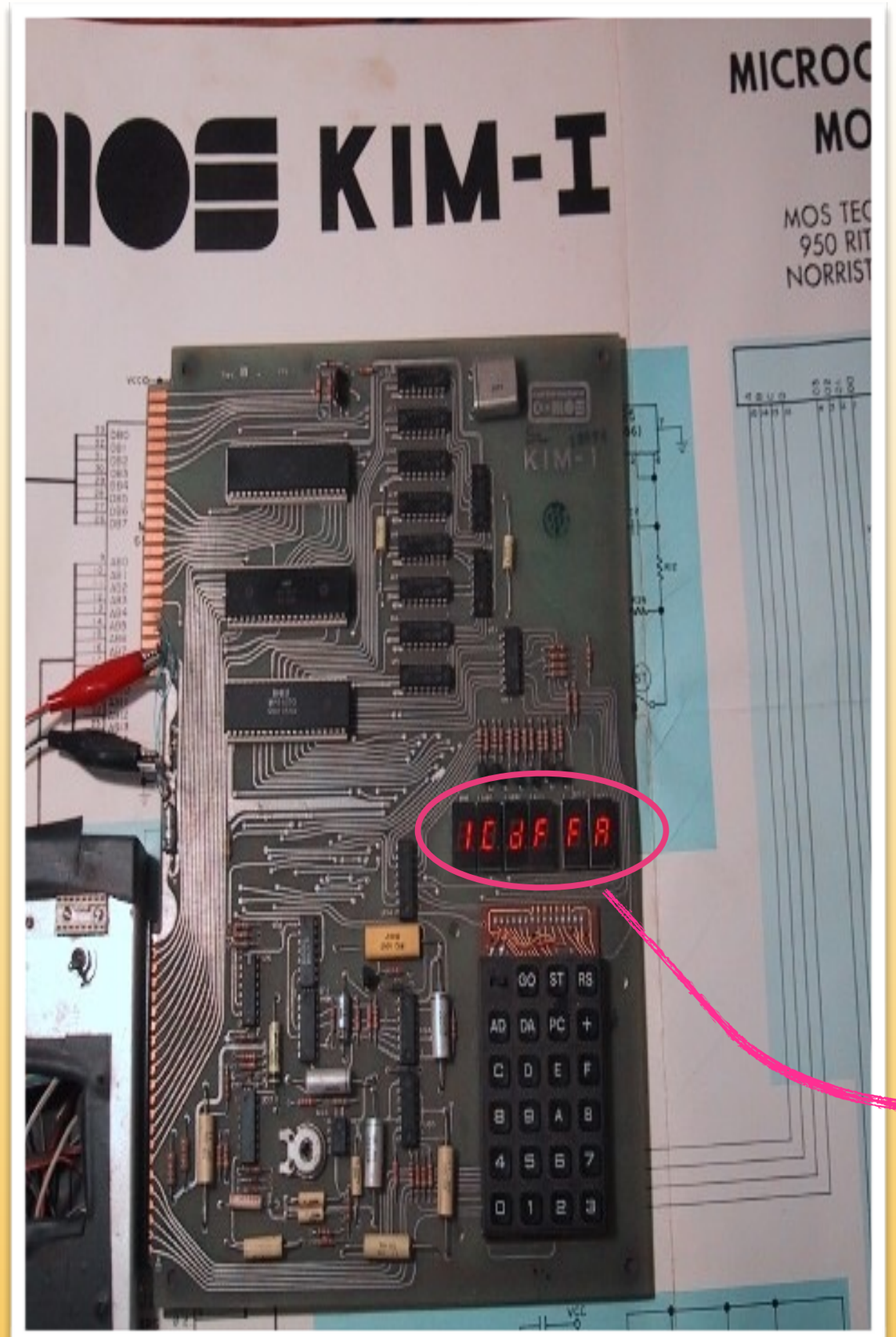
O “clássico" Apple II

Execução de Binários:



Apple II (1976)

Virtual][is a program that emulates the vintage Apple II computer on your Mac (http://www.virtualii.com)



Exemplo de kit com µprocessador 8-bits

Interface com usuário:

- **teclado hexadecimal**
- **6 x Displays 7-Segmentos (hexa)**
- **“Build Your Own KIM-1 with Ruud Baltissen” —> http://www.6502.org/trainers/buildkim/buildkim.htm**

Repare:

```
1CdF  FA
```

Endereço : Conteúdo

(Instrução ou Dado)

Diagrama elétrico do kit KIM-1 (exemplo)

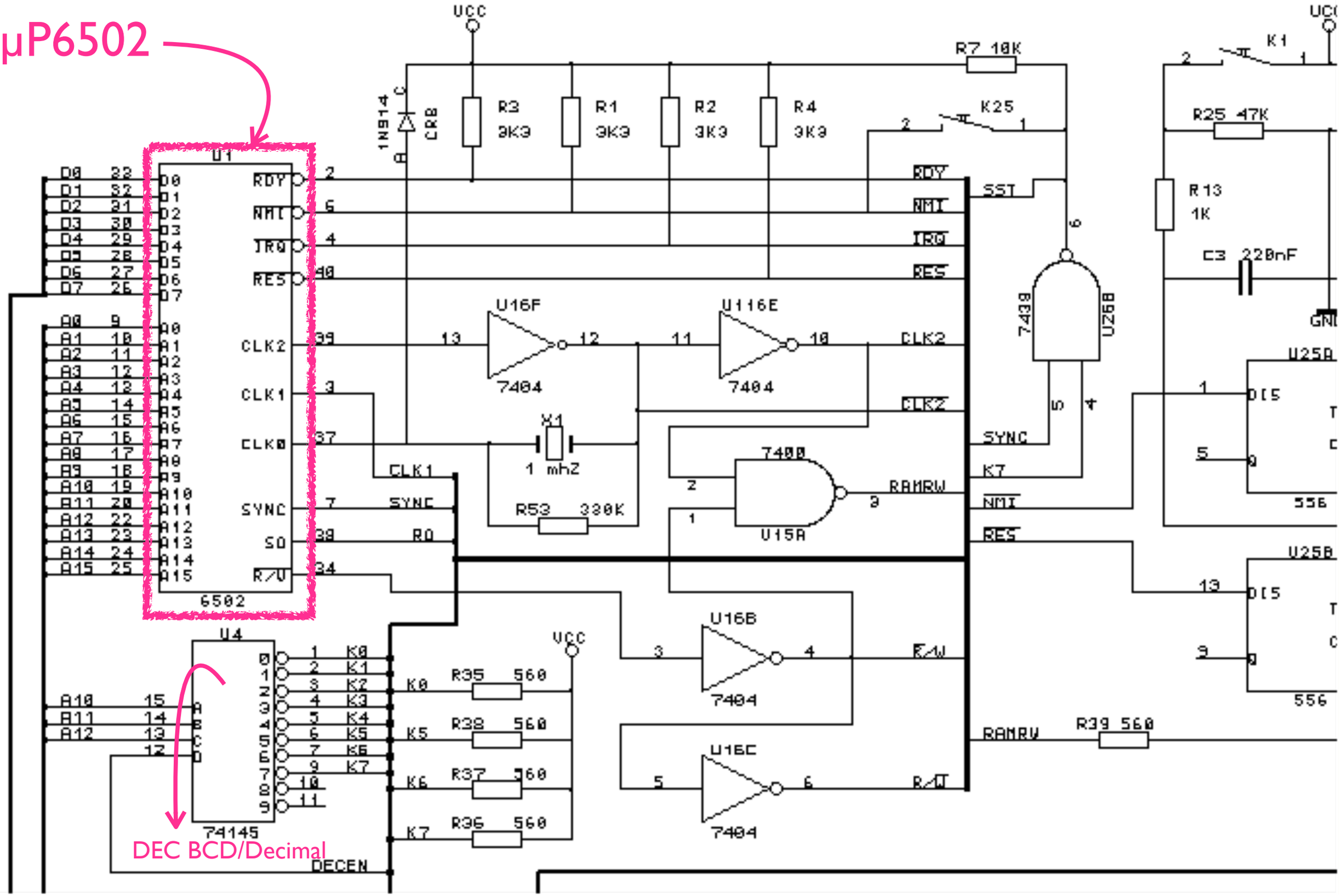
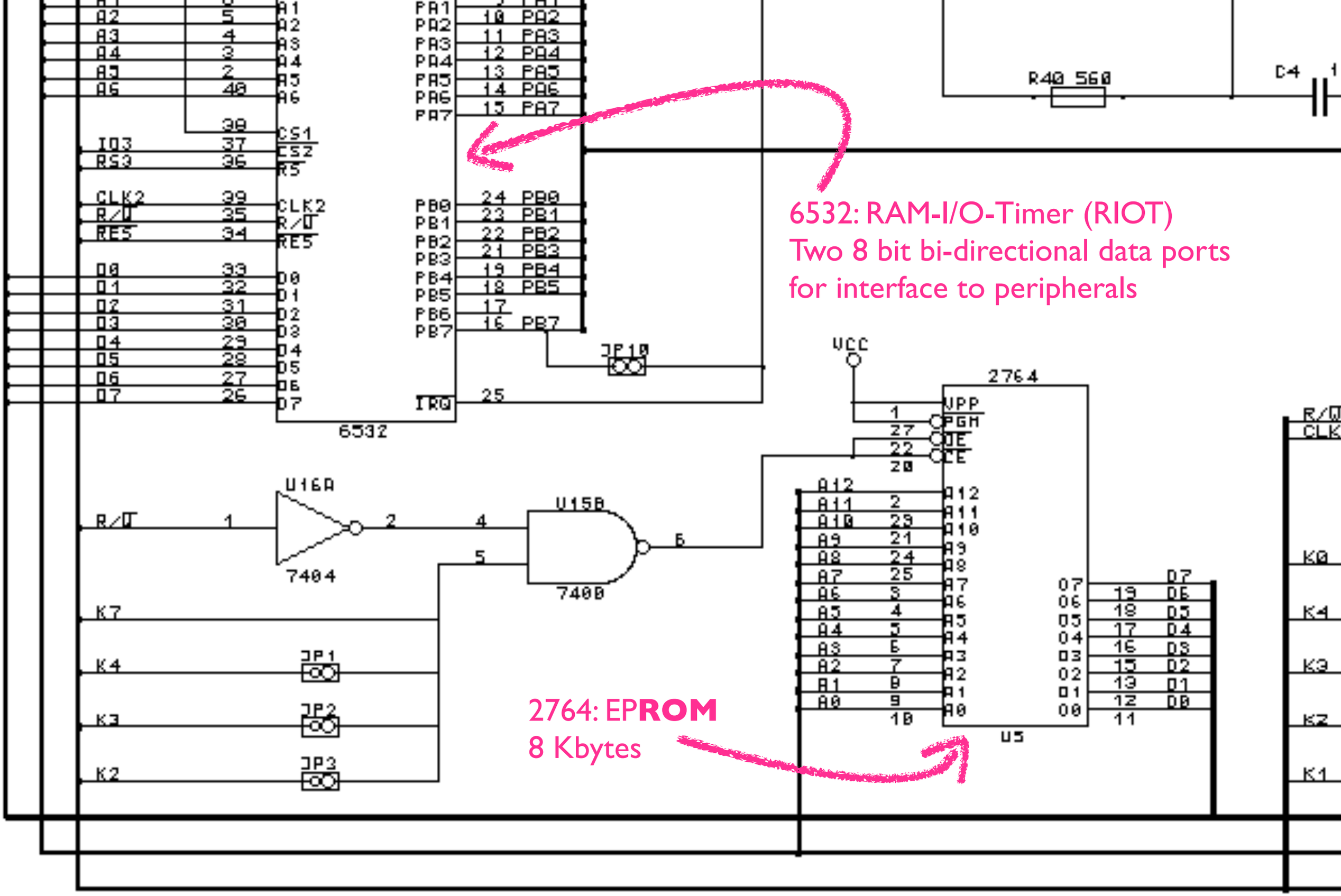
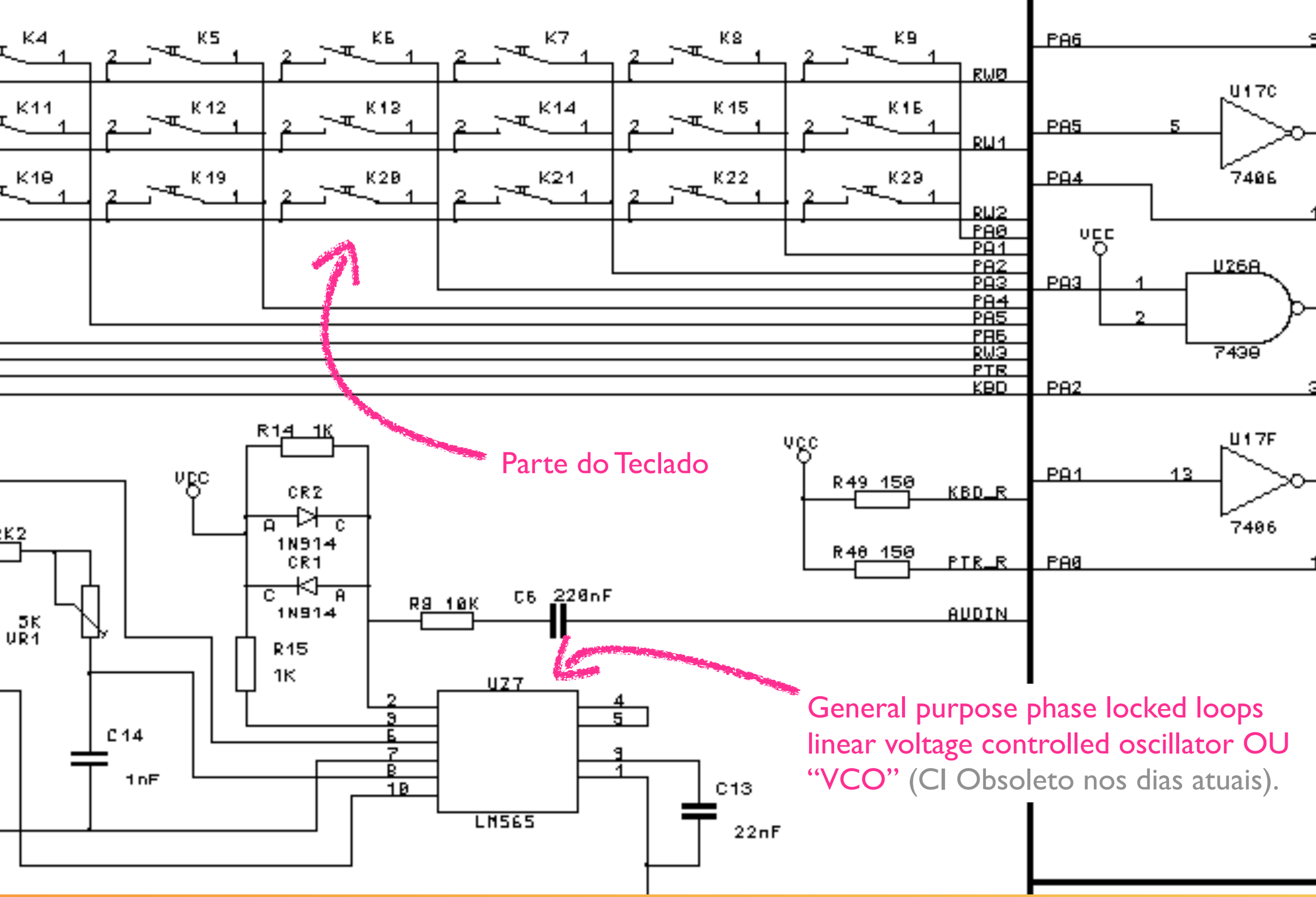


Diagrama elétrico do kit KIM-1 (exemplo)


6532: RAM-I/O-Timer (RIOT)
Two 8 bit bi-directional data ports
for interface to peripherals
2764: EPROM
8 Kbytes
6532
7404
7400
2764
U5
R40 560
C4
JP1
JP2
JP3
JP10



Parte do Teclado

General purpose phase locked loops linear voltage controlled oscillator OU "VCO" (CI Obsoleto nos dias atuais).

Displays multiplexados !

Parte do Teclado

Endereço do programa

Códigos Hexa

Linguagem Assembly

Exemplo de código:

```
0200 A2 EA     LDX SET NO. OF LOOPS FOR 1 SECOND
   2 CA        DEX
   3 A5 60     LDA STORE HOURS IN Fb
   5 85 Fb     STA
   7 A5 61     LDA STORE MIN'S IN FA
   9 85 FA     STA
   b A5 62     LDA STORE SEC'S IN F9
   d 85 F9     STA
   F 86 63     STX SAVE X
  11 84 64     STY (NOT NECESSARY, FILLER)       HR     MIN     SEC
  13 20 1F 1F "SCANDS" (DISPLAY TIME)          1 0    1 0     0 1
  16 A6 63     LDX                              Fb     FA      F9
  18 A4 64     LDY                            (0060) (0061) (006
  1A E0 00     CPX TO LOOP (TO 0202)
  1C d0 E4     BNE
  1E F8        SED SET DECIMAL MODE TO AVOID HEX DIGITS
  1F 38        SEC SET CARRY
  20 A9 00     LDA
  22 65 62     ADC ADD A+C+M-->A (0+1+SEC-->ACC.)
  24 85 62     STA STORE IN 62 (SEC) (ACC--> 62)
  26 d8        CLD CLEAR DECIMAL MODE FOR "SCANDS"
  27 C9 60     CMP TO LOOP (TO 0200) (RESETTING LOOP FOR NEW SEC
  29 d0 d5     BNE
  2b F8        SED
  2C 38        SEC SAME AS SECONDS
  2d A9 00     LDA
  2F 85 62     STA RESET SEC TO 00
  31 65 61     ADC ADD 0+1+MIN-->ACC
  33 85 61     STA STORE IN 61 (MIN) (ACC-->61)
  35 d8        CLD
  36 C9 60     CMP TO LOOP (TO 0200)
  38 d0 C6     BNE
  3A F8        SED SAME AS MINUTES
  3b 38        SEC
  3C A9 00     LDA
  3E 85 62     STA RESET SEC TO 00
  40 85 61     STA RESET MIN TO 00
  42 65 60     ADC ADD 0+1+HRS-->ACC
  44 85 60     STA
  46 d8        CLD                          FOR 24 HR CLOCK
  47 C9 13     CMP                            47 C9, 24
  49 d0 b5     BNE                            4b A9, 00
```

Exemplo de kit de microprocessador de 8-bits: "Kim" (*K*eyboard *I*nput *M*onitor)

Endereço do programa

Códigos Hexa

Linguagem Assembly

Exemplo de código:

```
0200 A2 EA    LDX SET NO. OF LOOPS FOR 1 SECOND
   2 CA       DEX
   3 A5 60    LDA STORE HOURS IN Fb
   5 85 Fb    STA
   7 A5 61    LDA STORE MIN'S IN FA
   9 85 FA    STA
   b A5 62    LDA STORE SEC'S IN F9
   d 85 F9    STA
   F 86 63    STX SAVE X
  11 84 64    STY (NOT NECESSARY, FILLER)      HR    MIN    SEC
  13 20 1F 1F "SCANDS" (DISPLAY TIME)         1 0    1 0    0 1
  16 A6 63    LDX                              Fb     FA     F9
  18 A4 64    LDY                            (0060) (0061) (006
  1A E0 00    CPX TO LOOP (TO 0202)
  1C d0 E4    BNE
  1E F8       SED SET DECIMAL MODE TO AVOID HEX DIGITS
  1F 38       SEC SET CARRY
  20 A9 00    LDA
  22 65 62    ADC ADD A+C+M-->A (0+1+SEC-->ACC.)
  24 85 62    STA STORE IN 62 (SEC) (ACC--> 62)
  26 d8       CLD CLEAR DECIMAL MODE FOR "SCANDS"
  27 C9 60    CMP TO LOOP (TO 0200) (RESETTING LOOP FOR NEW SEC
  29 d0 d5    BNE
  2b F8       SED
  2C 38       SEC SAME AS SECONDS
  2d A9 00    LDA
  2F 85 62    STA RESET SEC TO 00
  31 65 61    ADC ADD 0+1+MIN-->ACC
  33 85 61    STA STORE IN 61 (MIN) (ACC-->61)
  35 d8       CLD
  36 C9 60    CMP TO LOOP (TO 0200)
  38 d0 C6    BNE
  3A F8       SED SAME AS MINUTES
  3b 38       SEC
  3C A9 00    LDA
  3E 85 62    STA RESET SEC TO 00
  40 85 61    STA RESET MIN TO 00
  42 65 60    ADC ADD 0+1+HRS-->ACC
  44 85 60    STA
  46 d8       CLD                          FOR 24 HR CLOCK
  47 C9 13    CMP                            47 C9, 24
  49 d0 b5    BNE                            4b A9, 00
```

ENDEREÇO	CONTEÚDO
0200H	A2
0201H	EA
0202H	CA
0203H	A5
0204H	60
0205H	85
⋮	⋮

Exemplo de kit de microprocessador de 8-bits: "Kim" (*K*eyboard *I*nput *M*onitor)

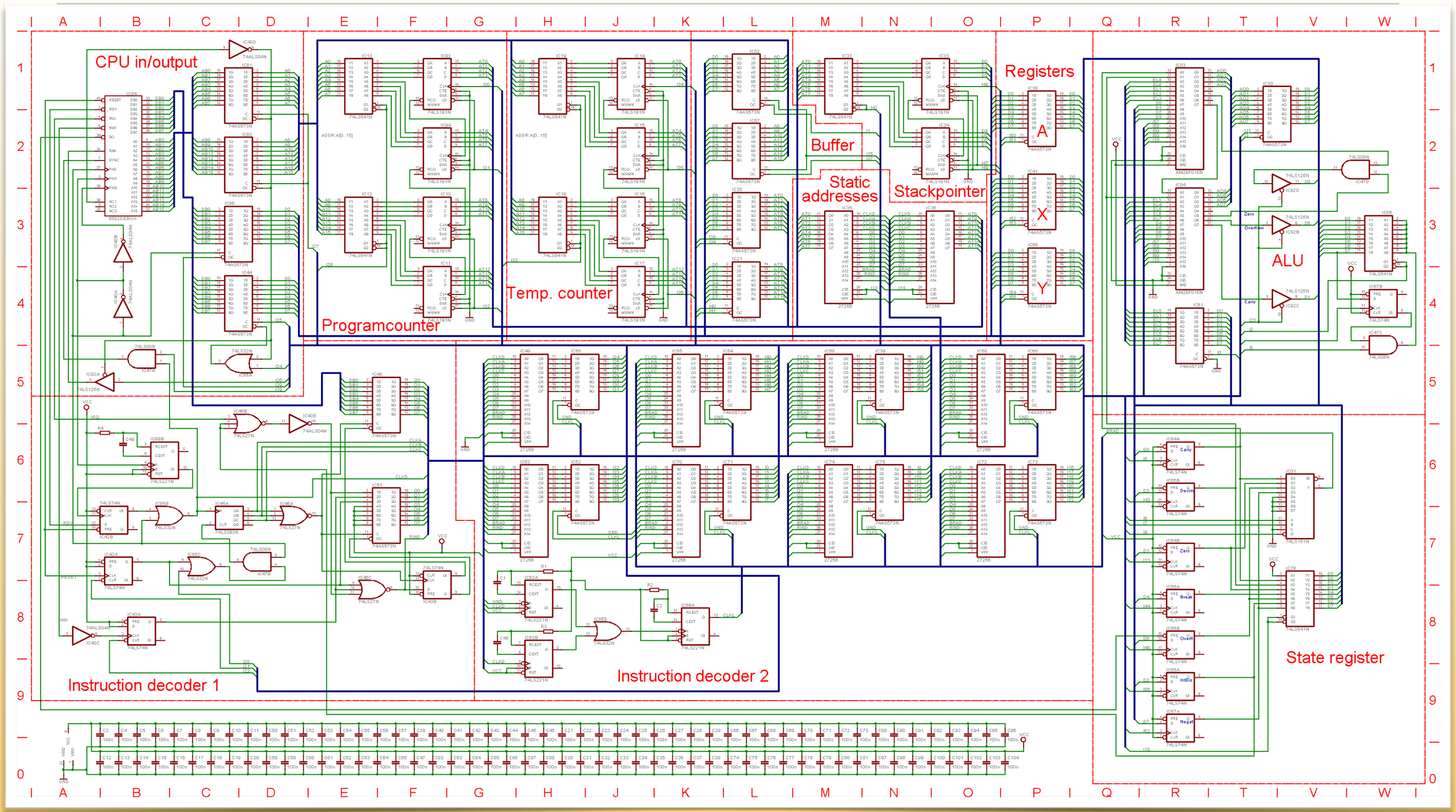


Diagrama elétrico do “Comodore" (PC de 8-bits, equipado com uP 6502c)

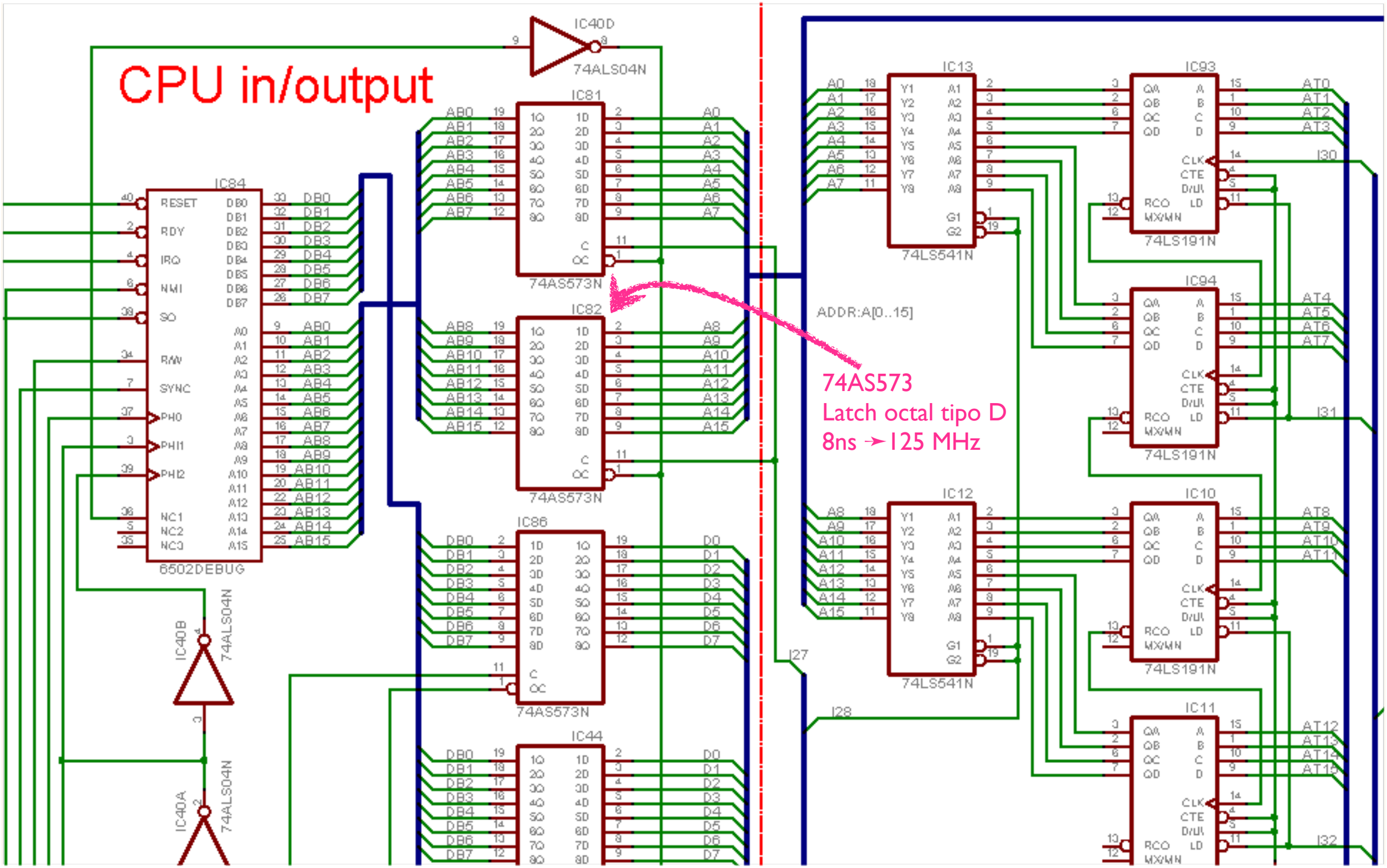
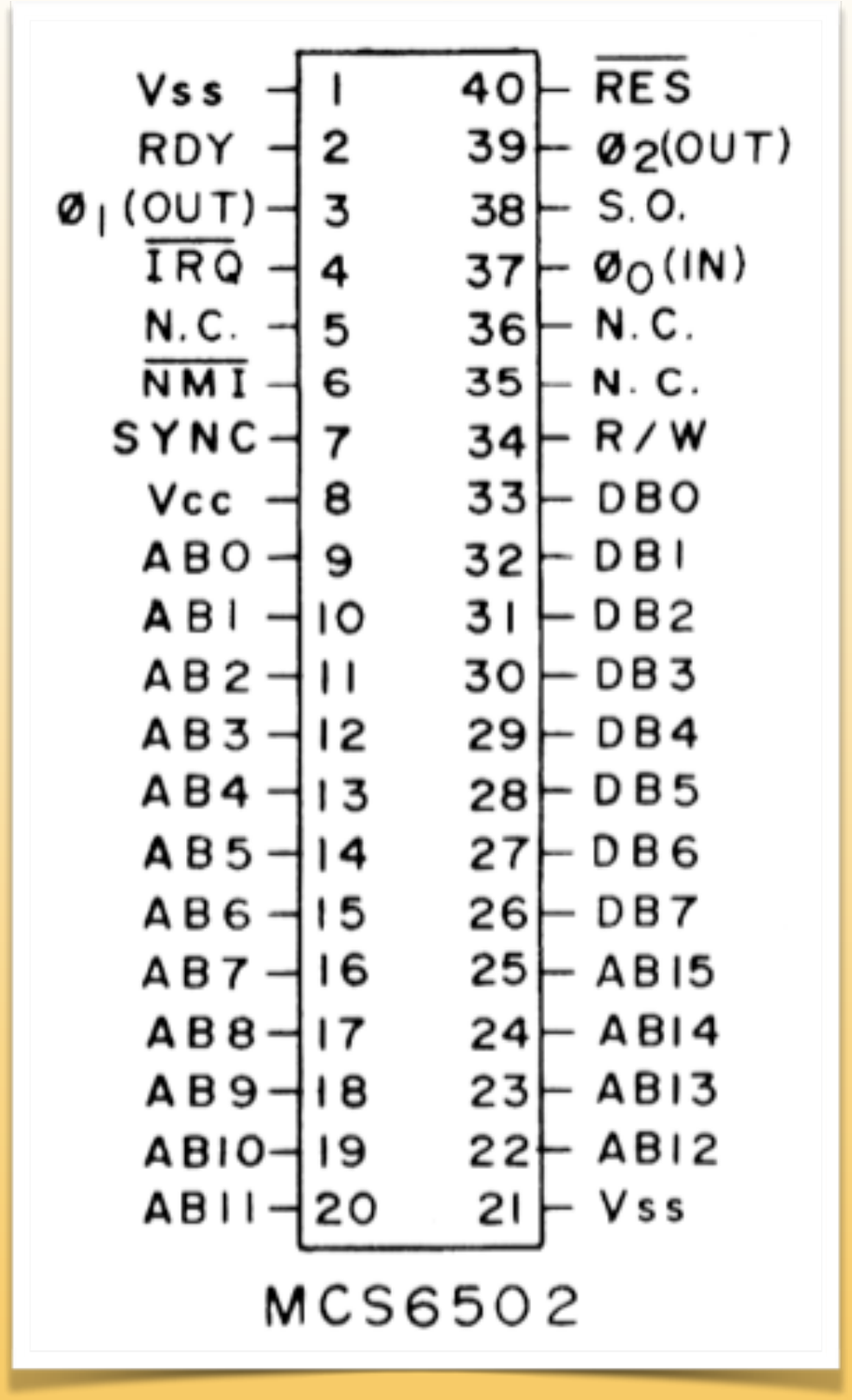


Diagrama elétrico do “Comodore" (PC de 8-bits, equipado com uP 6502c)

O μP 6502 (Aplle II)

- 8-bits;
- 4.528 transístores
  (Xeon µP: 2.500.000.000 transístores);



DIP 40 pinos, TTL, RC or Crystal Time Base Input[17]

Registradores, Flags

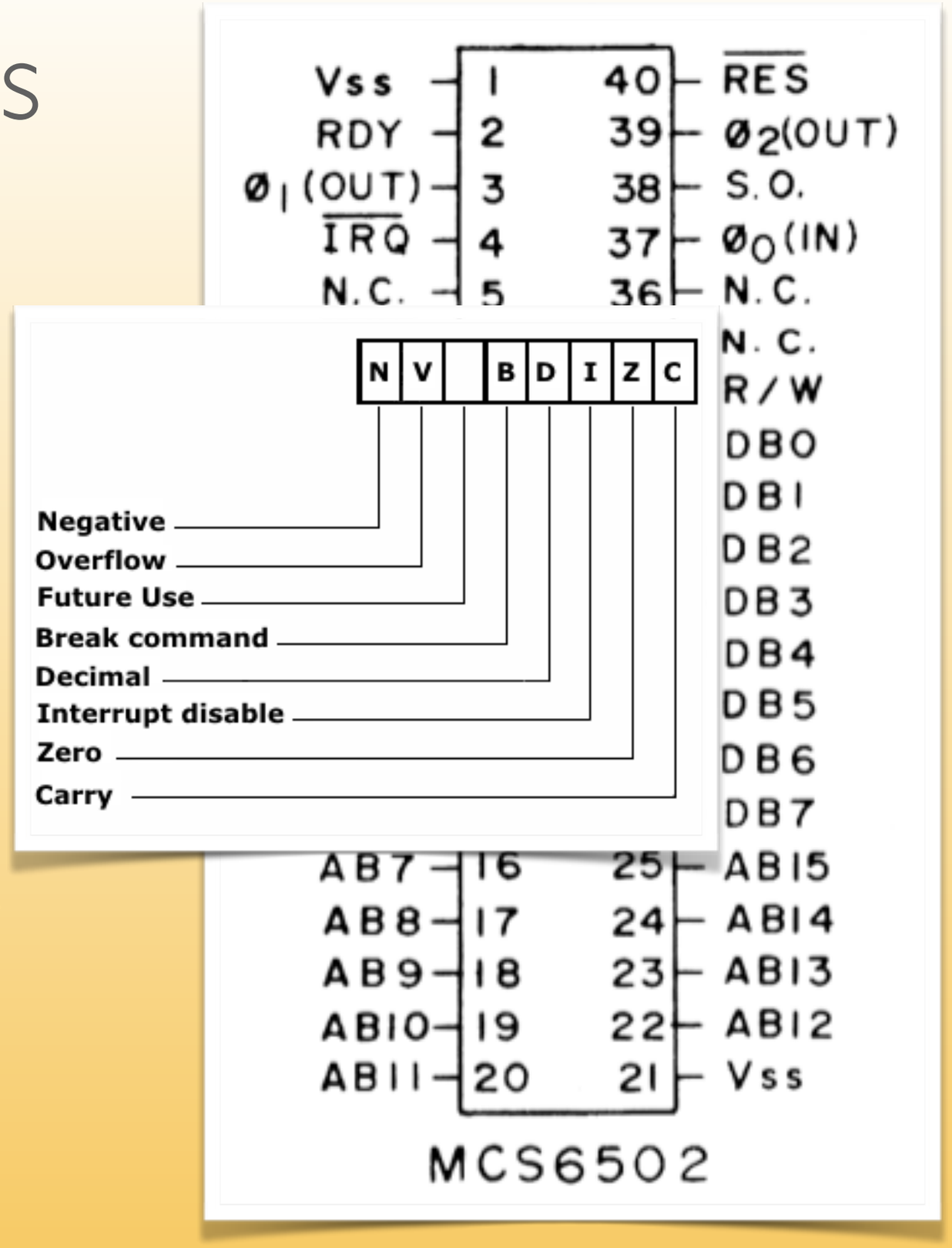
O µC 6502 inclui:

- registrador principal A (Acumulador);
- 2 x Registradores de Indexamento (X e Y);
- registrador Flag de status de 6 bits:

- registrador de pilha (stack pointer): S;
- registrador contador de programa, PC, de 16-bits (2^{16}=65.535 bytes);

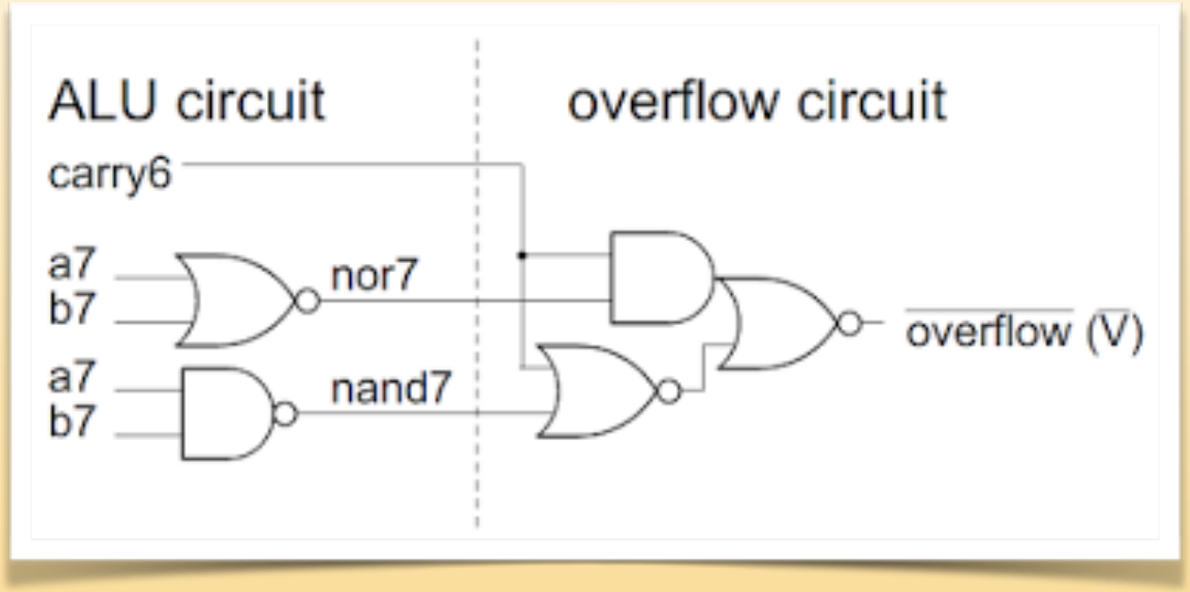
Existem implementações do µC 6502 em FPGA funcionando perfeitamente à 200 MHz (originalmente, 1978: 20 MHz). ➛http://iocoder.org/6502computer/ (Acessado Jun/2015)

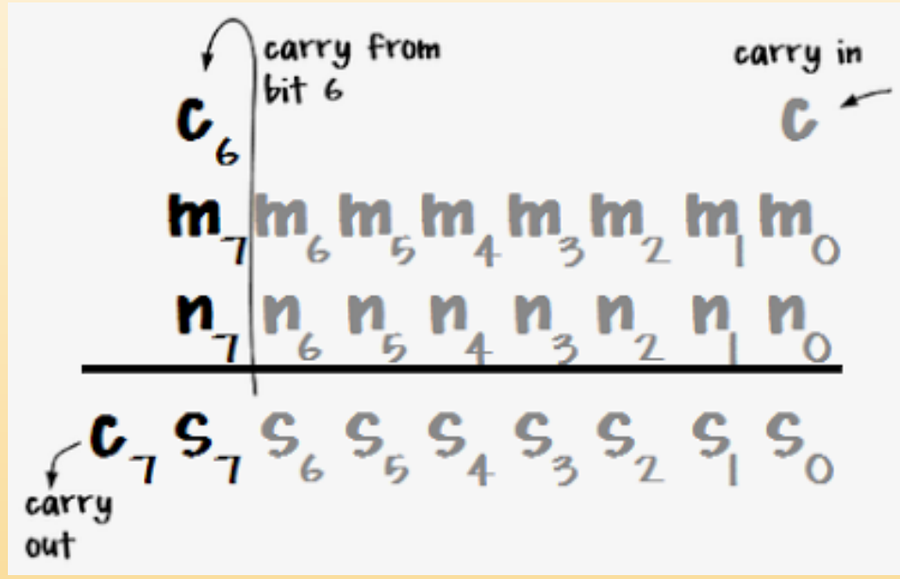
Grupo de Visual Transistor-level Simulation of the 6502 CPU and other chips! ➛ http://www.visual6502.org (Acessado Jun/2015)

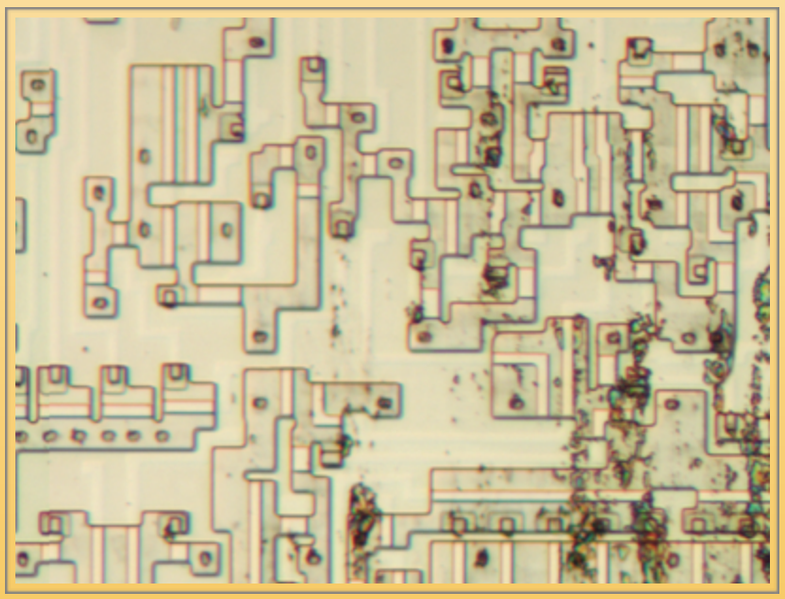


DIP 40 pinos, TTL, RC or Crystal Time Base Input

Overflow no µP6502 (8-bits)





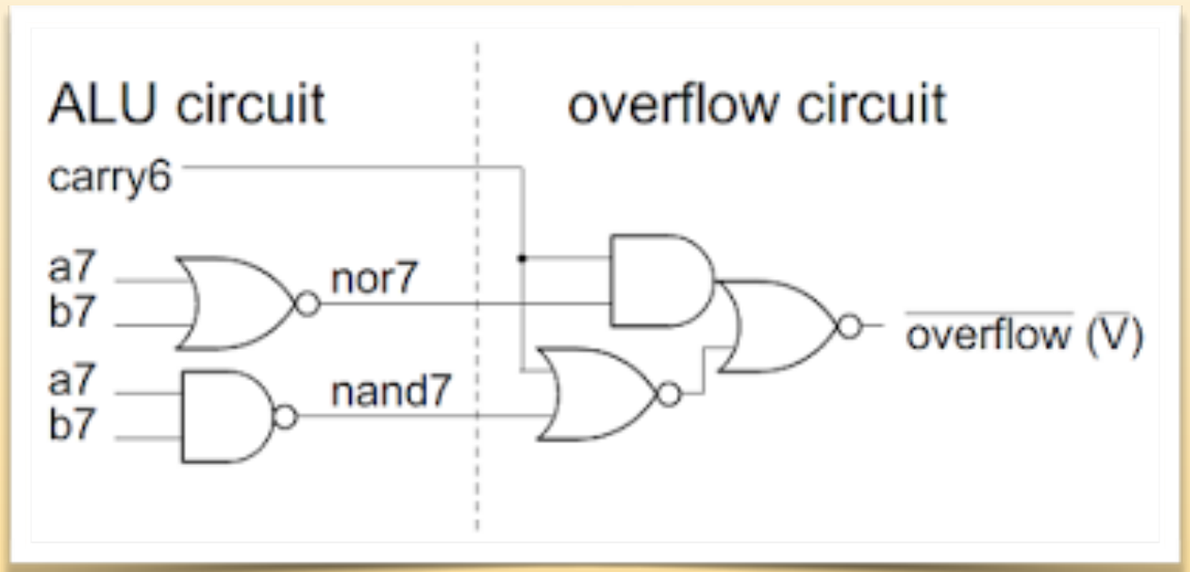


The diffusion layer of the 6502, zoomed in on the overflow circuit.

- 8 possibilidades.
- Para conjunto de 8-bits, há o bit de Carry-out (C7), o maior bit resultante de uma soma (S7) que também é o bit de sinal.
- O OVR ocorre para 4 possibilidades de argumentos envolvendo sinais (positivo+positivo, positivo+negativo, negativo+positivo, negativo+negativo) e deve prever ainda o carry-in (C).

Ref.: http://www.righto.com/2012/12/the-6502-overflow-flag-explained.html

Overflow no μP6502 (



- 8 possibilidades.
- Para conjunto de 8-bits, há o bit de Carry-out (C_7), o maior bit resultante de uma soma (S_7) que também é o bit de sinal.
- O OVR ocorre para 4 possibilidades de argumentos envolvendo sinais (positivo+positivo, positivo+negativo, negativo+positivo, negativo+negativo) e deve prever ainda o carry-in (C): total ➛ 8 combinações.

- Definição mais comum:

$$V = C_6 \oplus C_7$$

- Uma segunda equação leva em conta se os bits de sinal (M_7 e N_7) estão em 0 e o carry-in está em 1, ou se os bits de sinal estão em 1 e o carry-in em 0:

$$V = (\overline{M}_7 \cdot \overline{N}_7 \cdot C_6) + (M_7 \cdot N_7 \cdot \overline{C}_6)$$

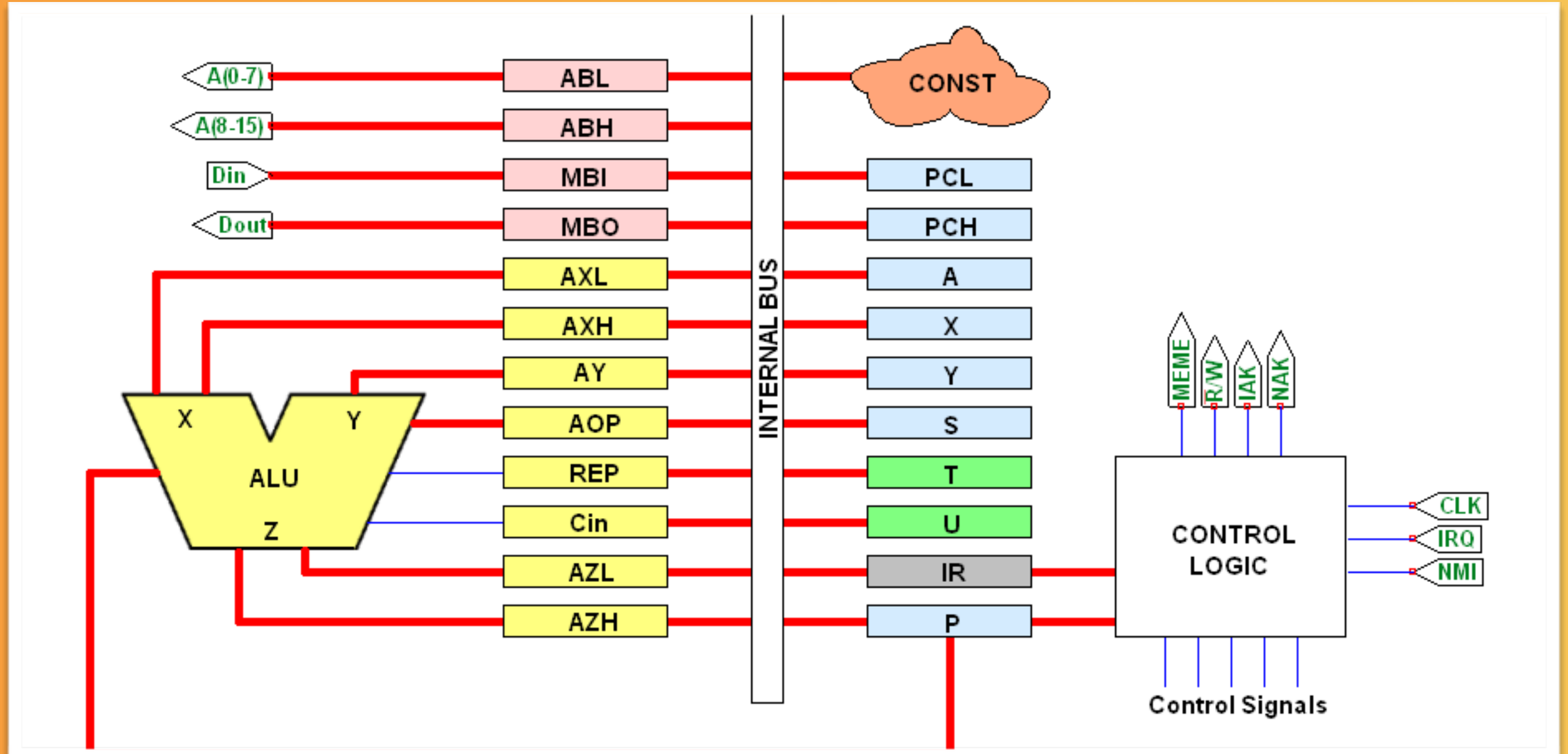
- Manipulando a equação usando De Morgan:

$$\overline{V} = \overline{\left[(\overline{M_7 + N_7}) \cdot C_6\right] + \left[\overline{(\overline{M_7 \cdot N_7}) + C_6}\right]}$$

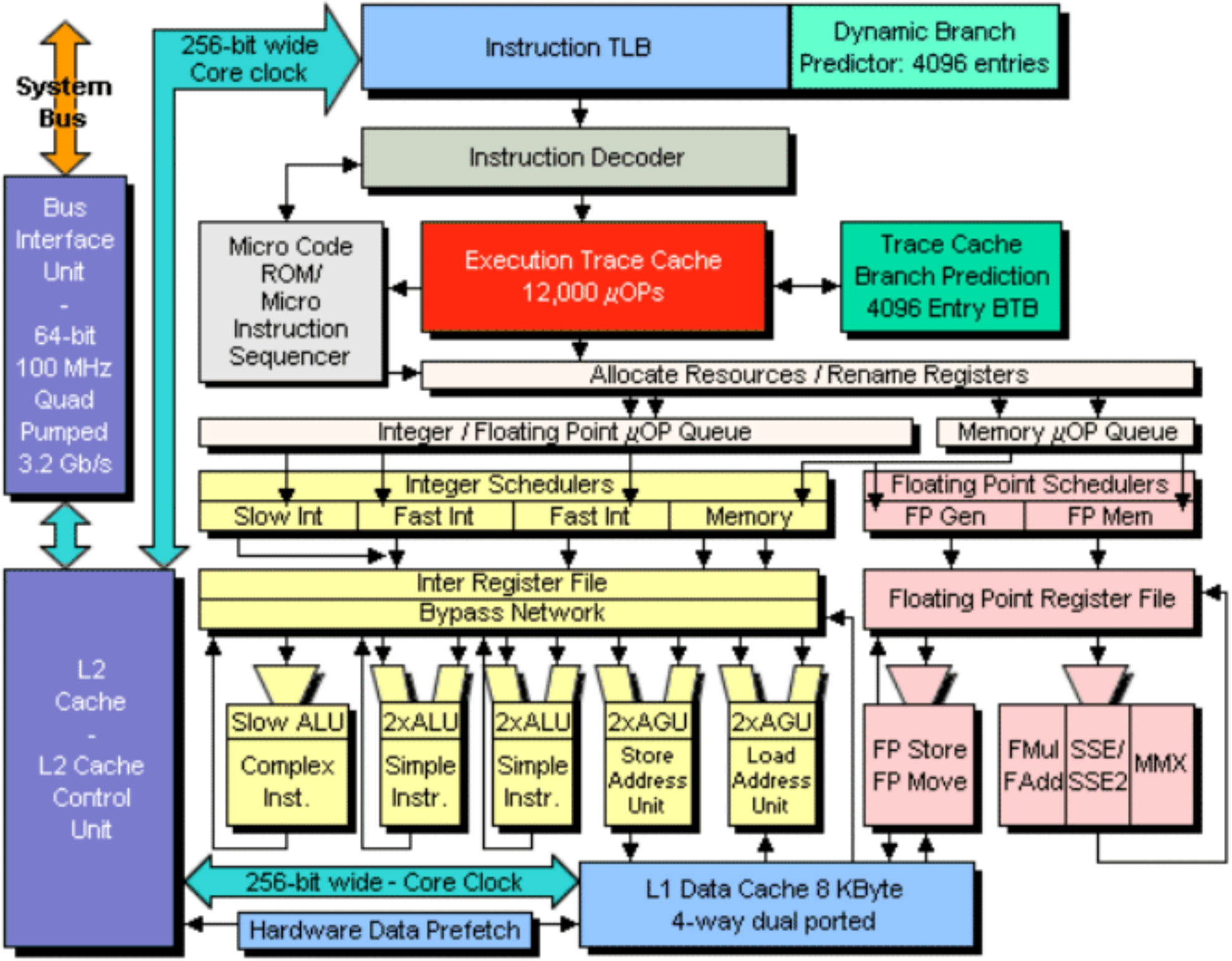
Ref.: http://www.righto.com/2012/12/the-6502-overflow-flag-explained.html

Operações que afetam o bit de OverFlow

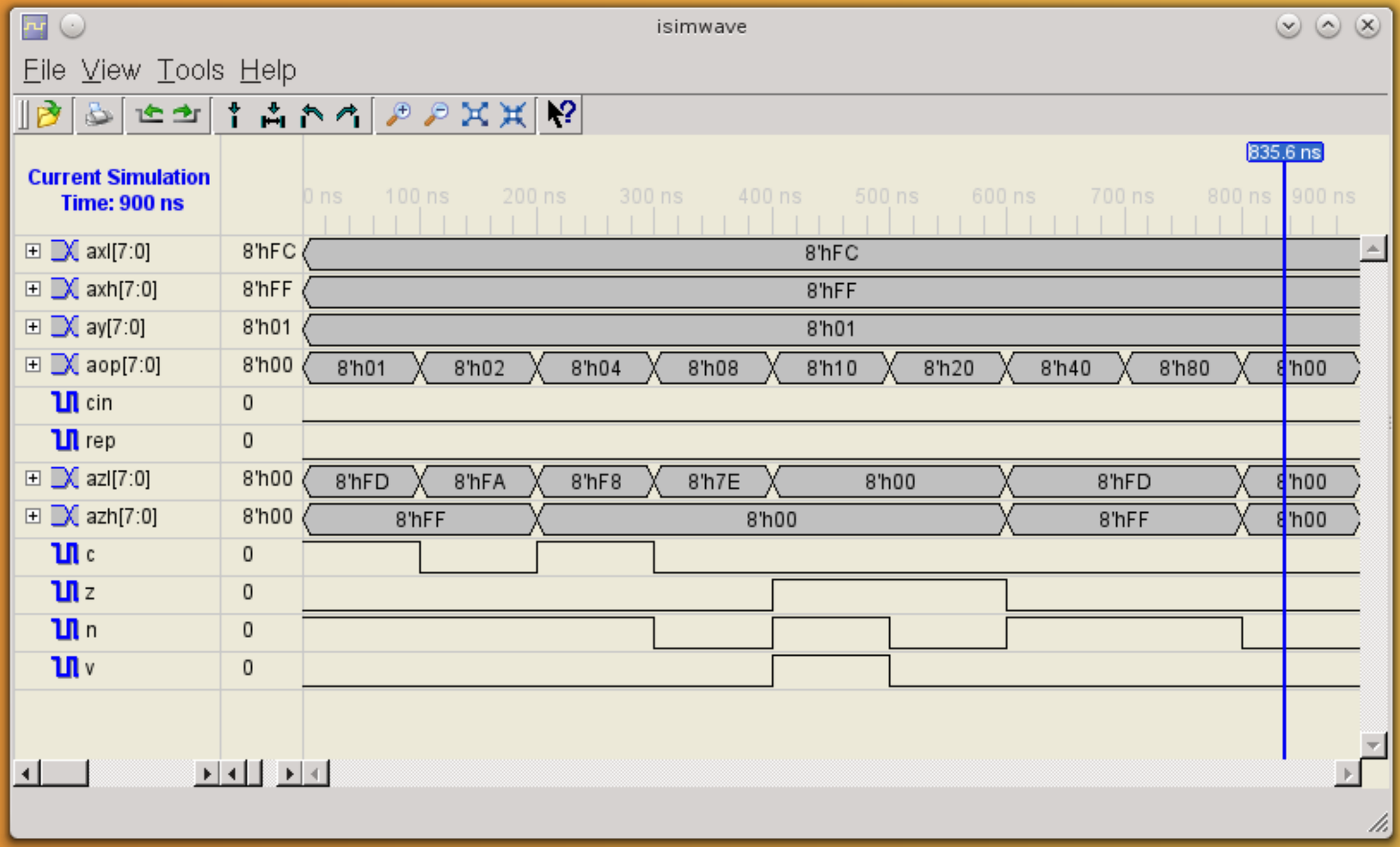
- Existem apenas 6 instruções (Assembly) que afetam o bit V: ADD, BIT, CLV, PLP, RTI e SBC.
- CLV = Clear V;
- PLP e RTI atualiza valores do registrador de Status (Flags) à partir dos valores do registrador de Pilha (Stack), afetando o bit V;
- BIT: relacionado com instruções de salto (Branch-If-True);
- ADC: adição e SBC: subtração. Os resultados destas operações ficam no Acumulador.
- 1 Byte: 256 combinações ➔ $00 à $FF (números sem sinal)



Arquitetura interna do µC6502

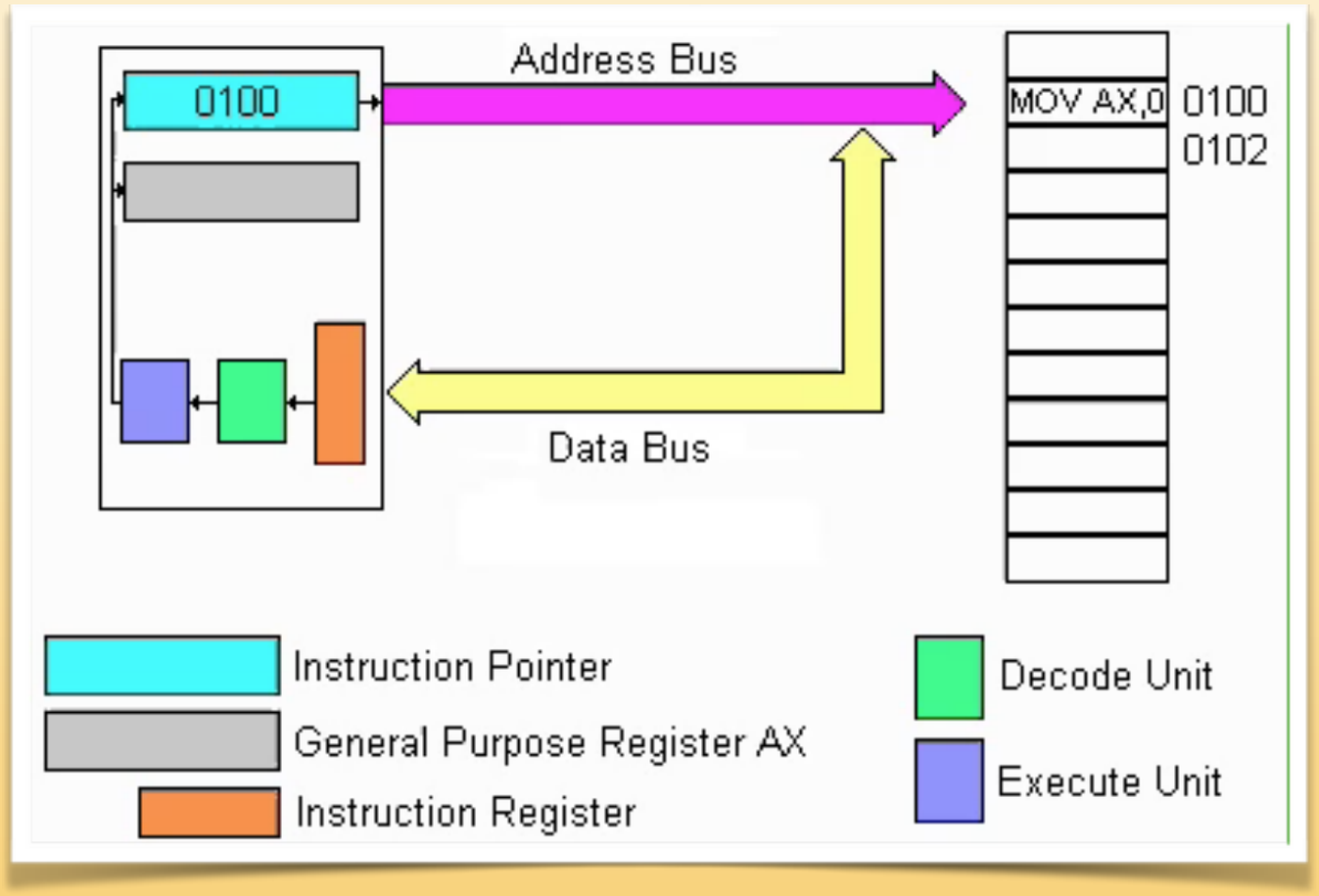


Arquitetura interna Pentium IV

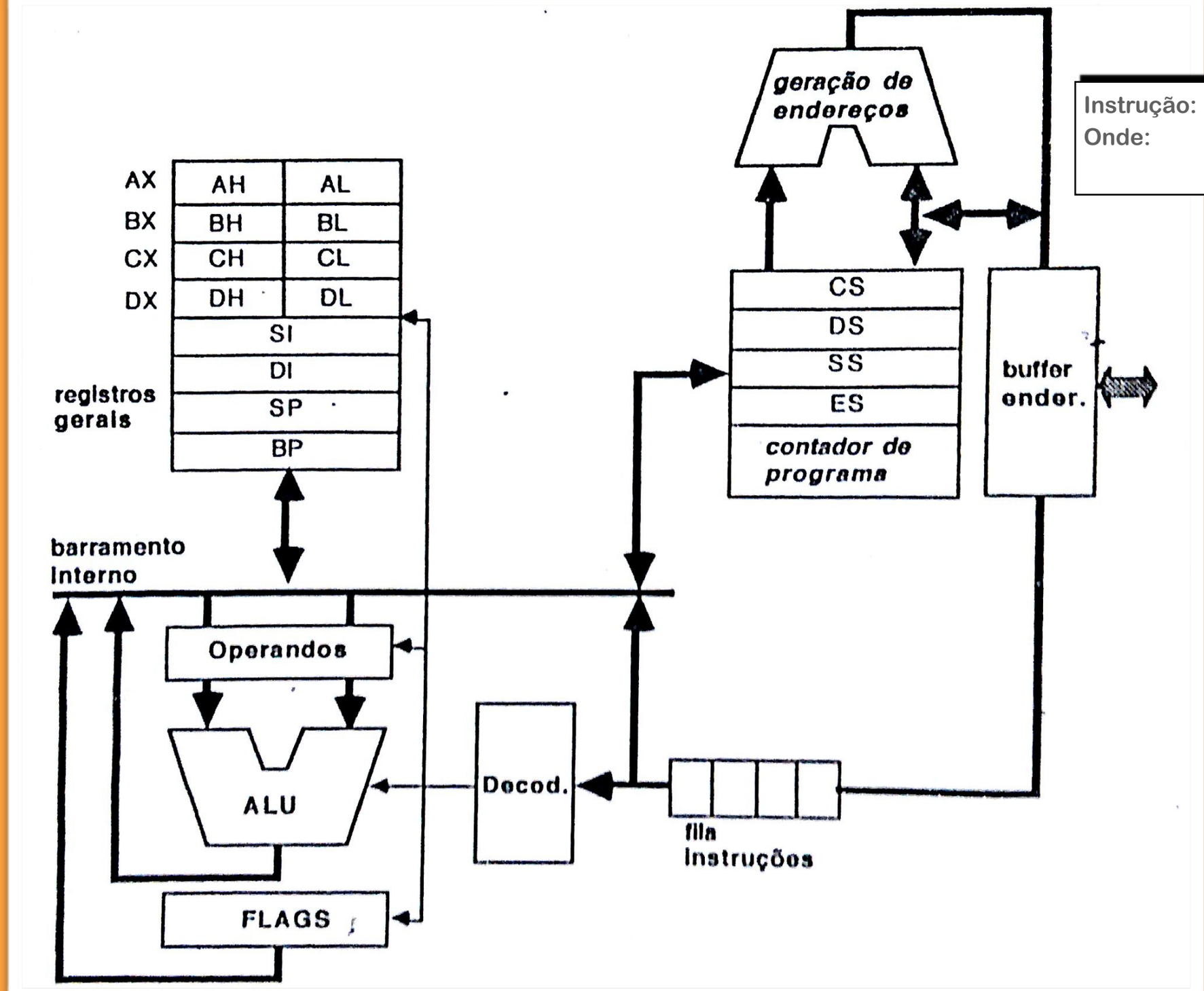


Simulação da ULA do µC6502

Ciclo de Busca e Execução de uma Instrução (Fetch & Execute)

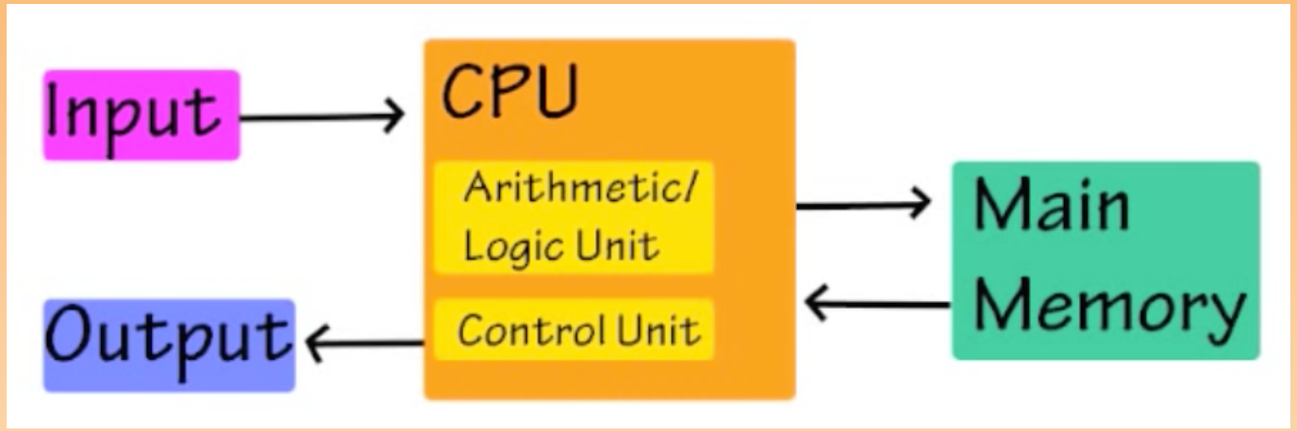


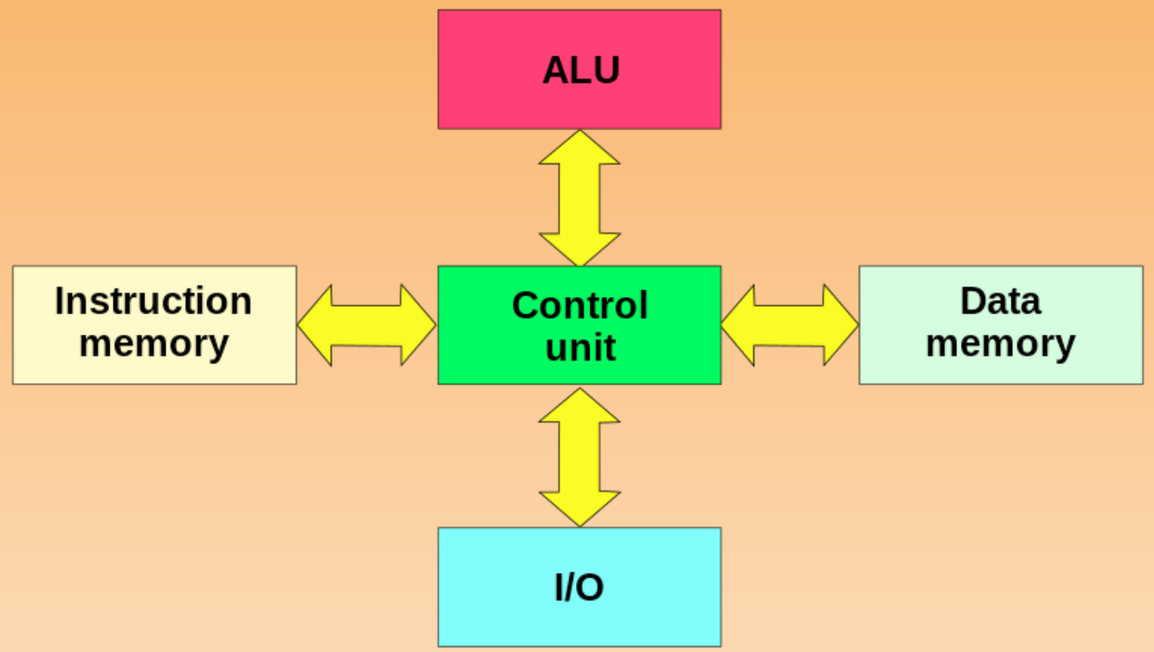

```
while running
   fetch instruction

   increment program counter (PC)

   decode instruction

   while  operands needed
     fetch operands
     increment PC

   endwhile

   execute instruction

end-while
```



Instrução: ADD AX,[DS:DI]
Onde: ADD op1, op2 faz: op1=op1+op2;
AX, DS e DI são registradores gerais.

Estrutura interna do 8086

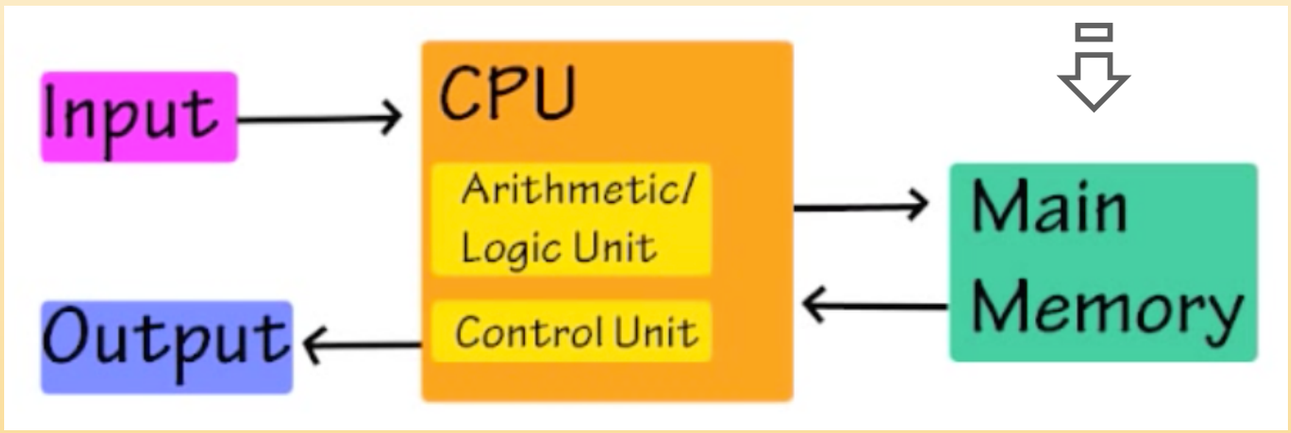


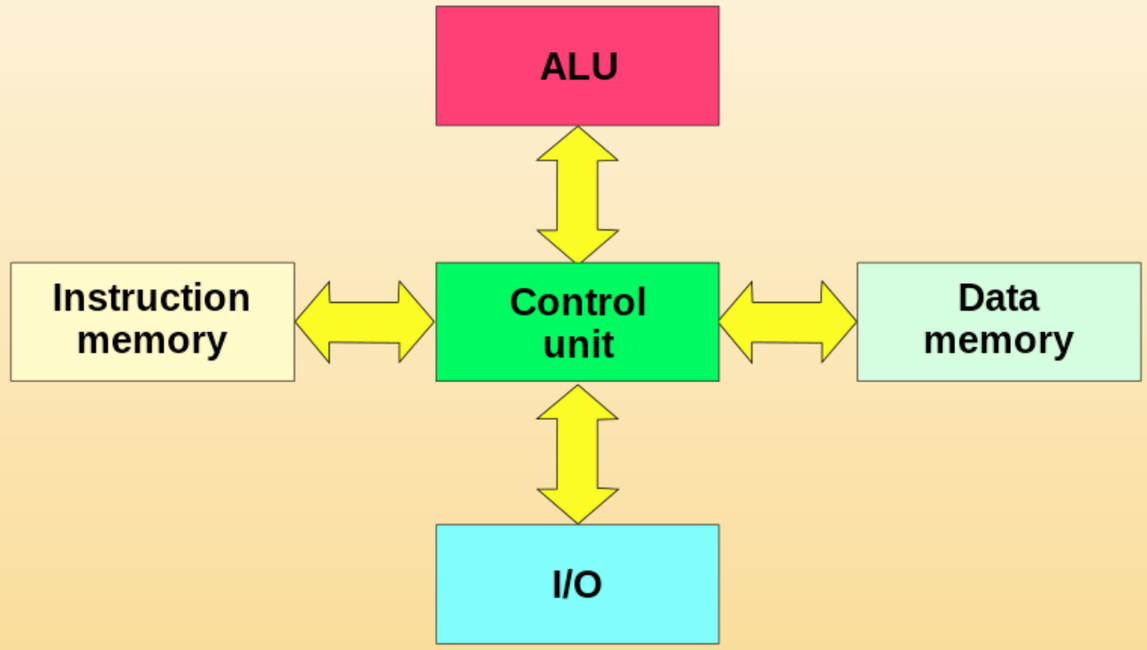


Arquitetura típicas de computadores

Modelo Von Neumann x Harward

Arquiteturas
Von Neumann x Harward



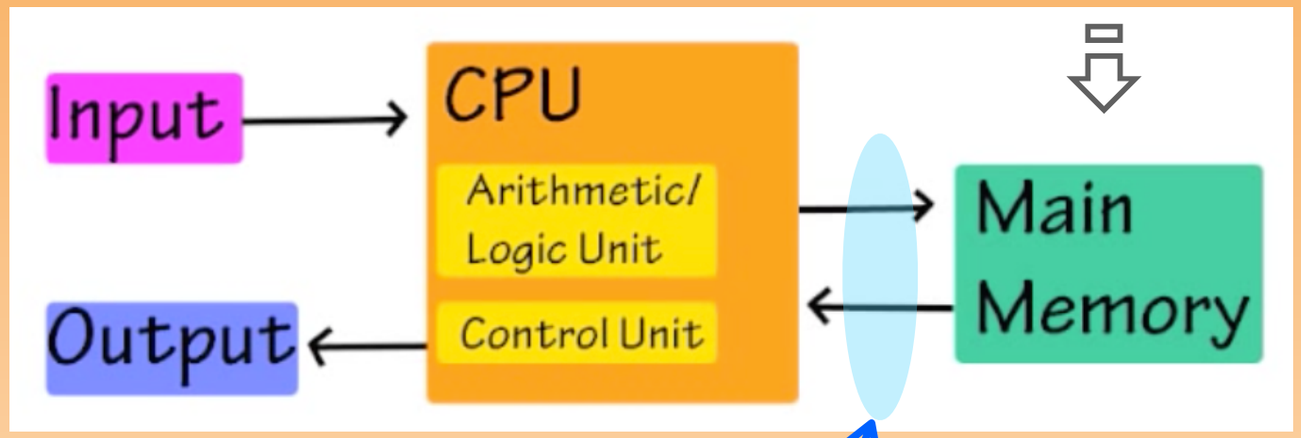


Mais antigo: 1945...
Aqui a máquina armazena seus programas no mesmo espaço de memória de seus dados;
Adotada na maioria dos uP's;

Mais recente;
Aqui, a máquina possui 2 memórias diferentes e independentes em termos de barramento e ligação ao processador (o acesso à memória de dados de modo separado em relação à memória de programa)
Prioridade: colocar o uP para trabalhar + rápido;
Adotada em microcontroladores como: PICs, DSPs, ARMs.

Von Neumann x Harward

Dados
+
Programas
(misturados)

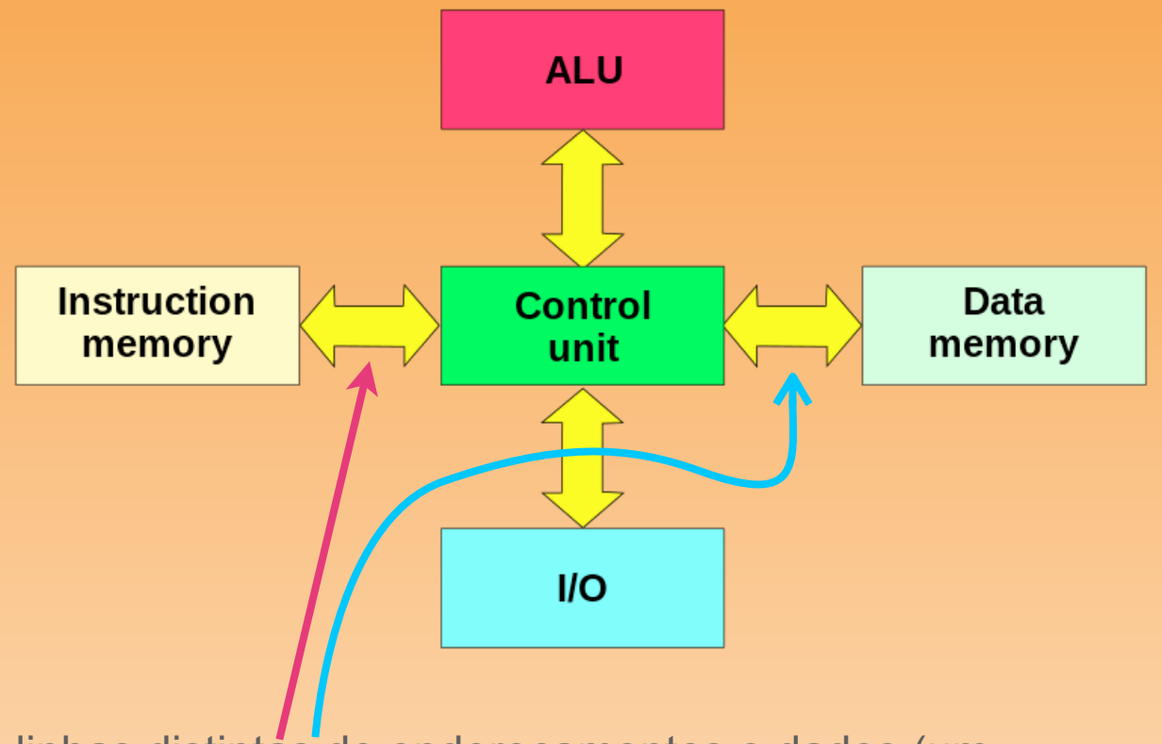


Desvantagem:

barramento comum de endereços e de dados.

É processada uma única informação por vez, visto que, nessa tecnologia, execução e dados percorrem o mesmo barramento, o que torna o processo lento em relação à arquitetura Harvard.

Usam memória cache para tentar melhorar a velocidade.



linhas distintas de endereçamentos e dados (um conjunto para programa e outro conjunto para dados);
Obs.: as larguras podem ser diferentes!

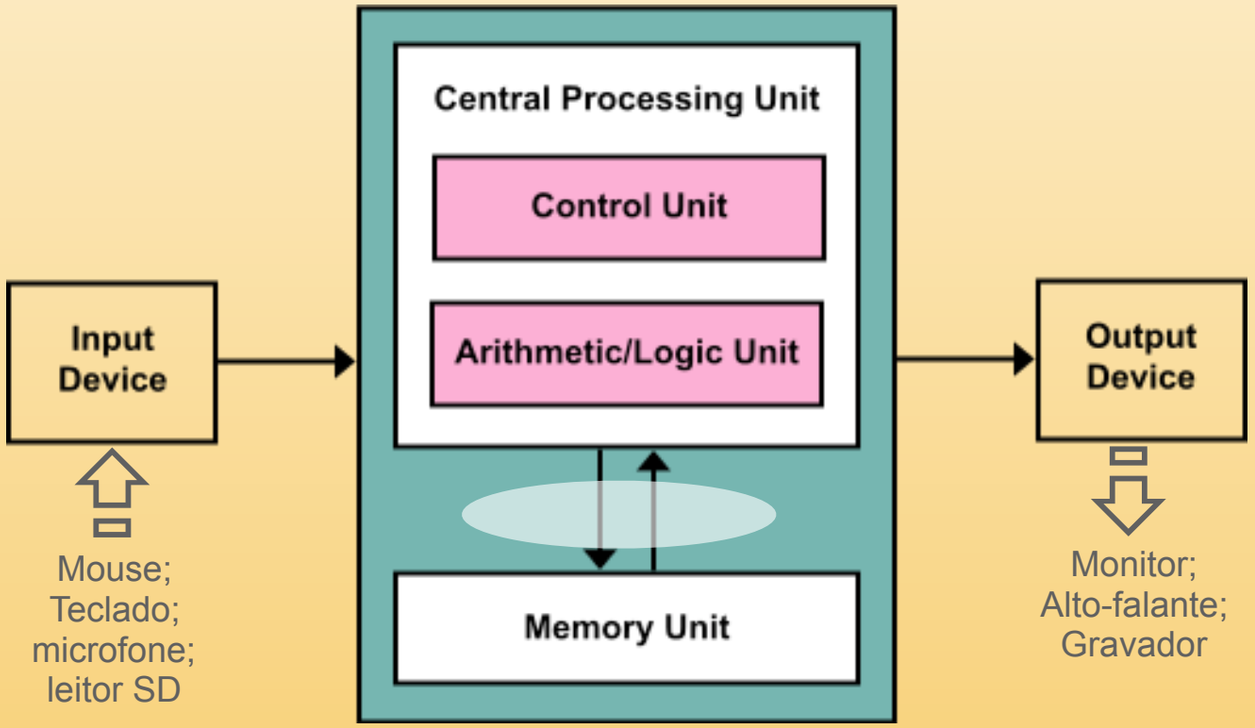
Característica principal:
Proporciona maior velocidade de processamento, pois, enquanto a CPU processa uma informação, outra nova informação está sendo buscada, de forma sucessiva.

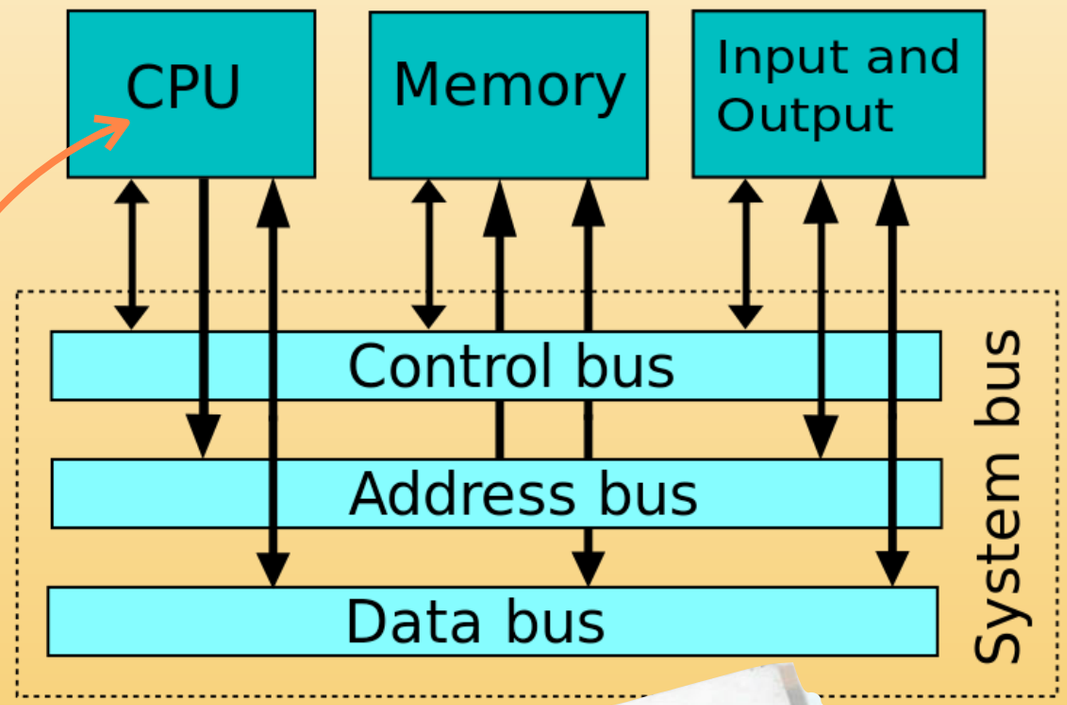
O uso da arquitetura de Harvard tem espaço de código e de dados, distintos: o endereço da instrução zero não é o mesmo que o endereço zero de dados. A instrução no endereço zero pode identificar um valor de 24 bits, enquanto o dado no endereço zero pode indicar um byte sem relação alguma com o conjunto de 24 bits das instruções!

Ciclo de:

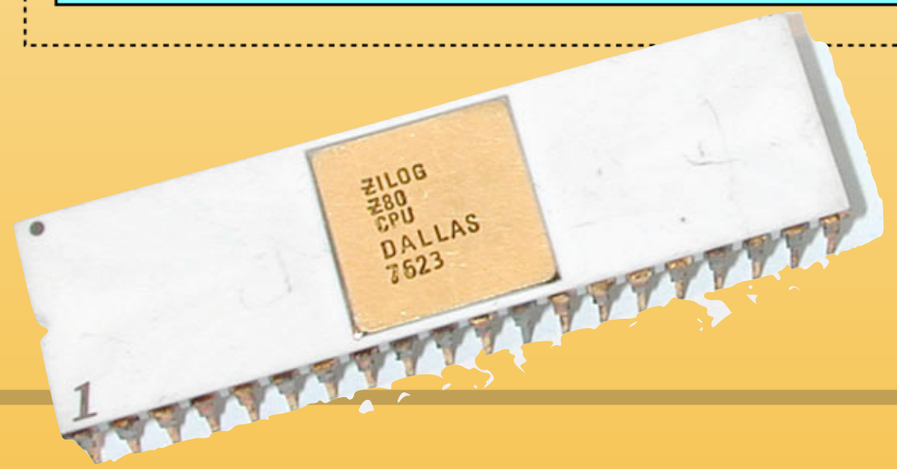
Arquitetura Von Neumann:

Dados + Programas
(misturado)

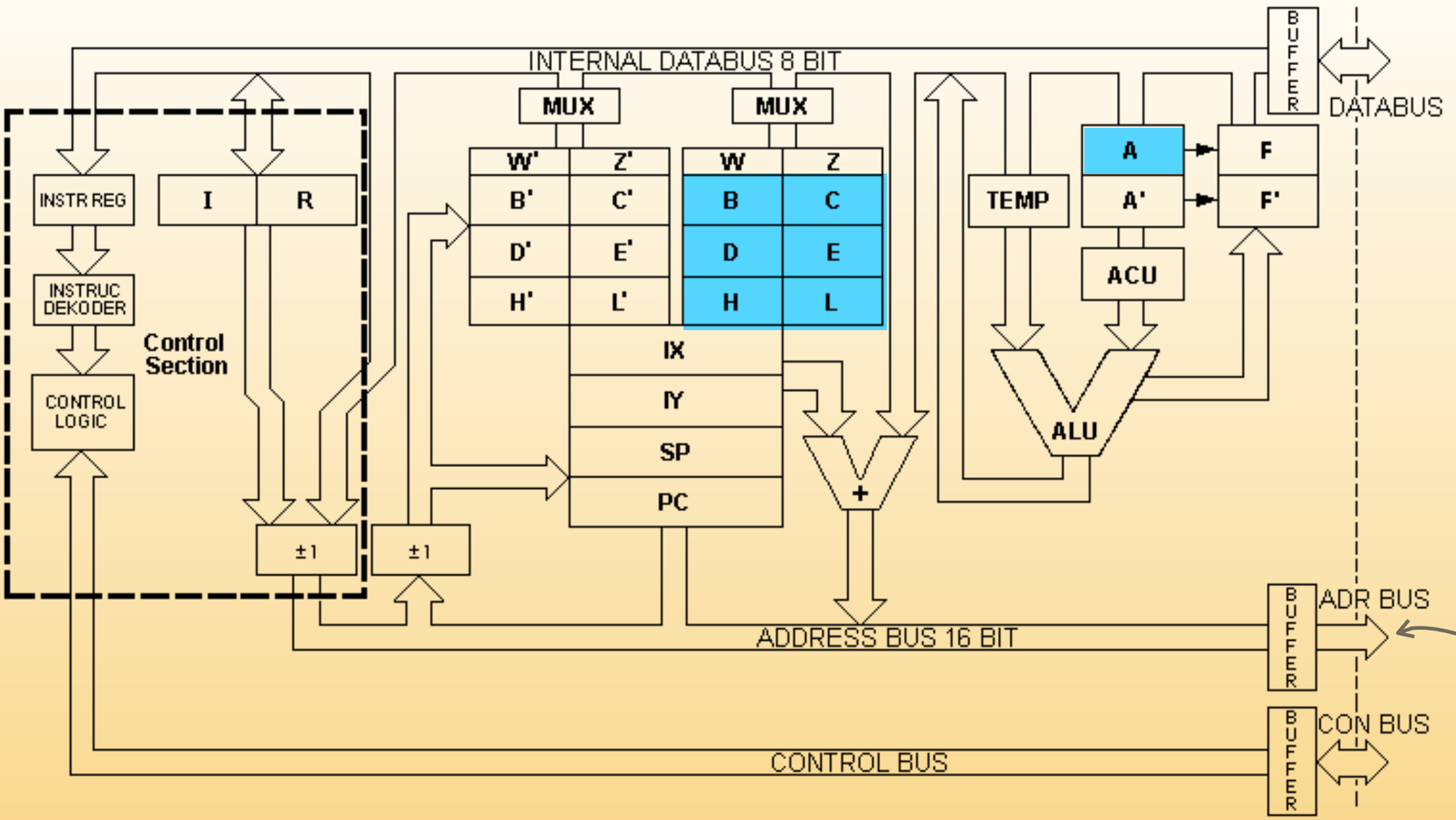


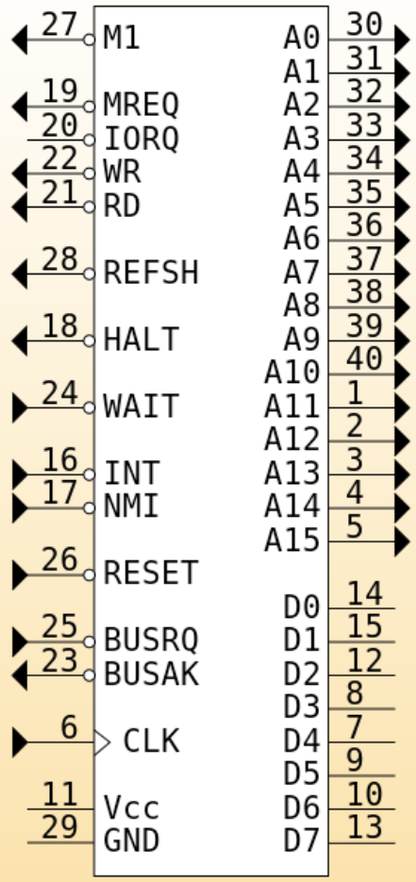


Necessitamos "eleger" um uP:
- Digamos que seja o Z80...



Estrutura interna do Z80:





Registradores:

A = Acumulador;
BC e DE = "genéricos";
HL = instruções envolvem endereçamento;

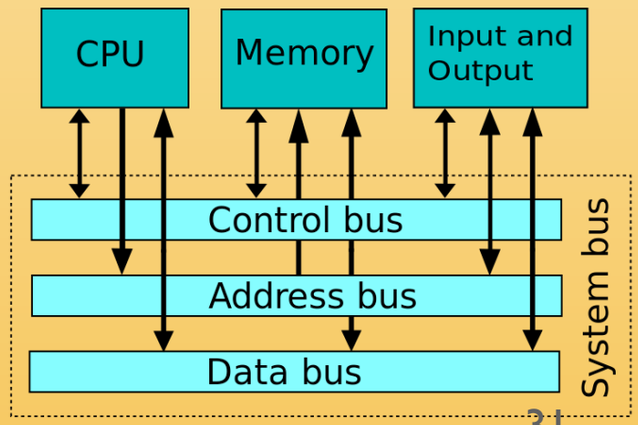
SP = Stack Pointer (ender. pilha);
PC = Program Counter (ender. programa);

IX, IY = para indexação (vetores). Ex. A[0], A[12],..

F = Flags:

S	Z	-	H	-	P	N	C

S=Sinal;
Z=Zero;
H=Half-carry;
P=Paridade
C=Carry.



- Exemplos de instruções de máquina (Assembly) do Z80:

Datapoint 2200 & i8008	i8080	Z80	i8086/i8088
before ~1973	~1974	1976	1978
LBC	MOV B,C	LD B,C	MOV BL,CL
--	LDAX B	LD A,(BC)	MOV AL,[BX]
LAM	MOV A,M	LD A,(HL)	MOV AL,[BP]
LBM	MOV B,M	LD B,(HL)	MOV BL,[BP]
--	STAX D	LD (DE),A	MOV [DX],AL [31]
LMA	MOV M,A	LD (HL),A	MOV [BP],AL
LMC	MOV M,C	LD (HL),C	MOV [BP],CL
LDI 56	MVI D,56	LD D,56	MOV DL,56
LMI 56	MVI M,56	LD (HL),56	MOV byte ptr [BP],56
--	LDA 1234	LD A,(1234)	MOV AL,[1234]
--	STA 1234	LD (1234),A	MOV [1234],AL
--	--	LD B,(IX+56)	MOV BL,[SI+56]
--	--	LD (IX+56),C	MOV [SI+56],CL
--	--	LD (IY+56),78	MOV byte ptr [DI+56],78
--	LXI B,1234	LD BC,1234	MOV BX,1234
--	LXI H,1234	LD HL,1234	MOV BP,1234
--	SHLD 1234	LD (1234),HL	MOV [1234],BP
--	LHLD 1234	LD HL,(1234)	MOV BP,[1234]
--	--	LD BC,(1234)	MOV BX,[1234]
--	--	LD IX,(1234)	MOV SI,[1234]

LOAD
STORE
MOVE
→ mnemônicos →
LD
STA
MOV, MVI
INC
ADD
...

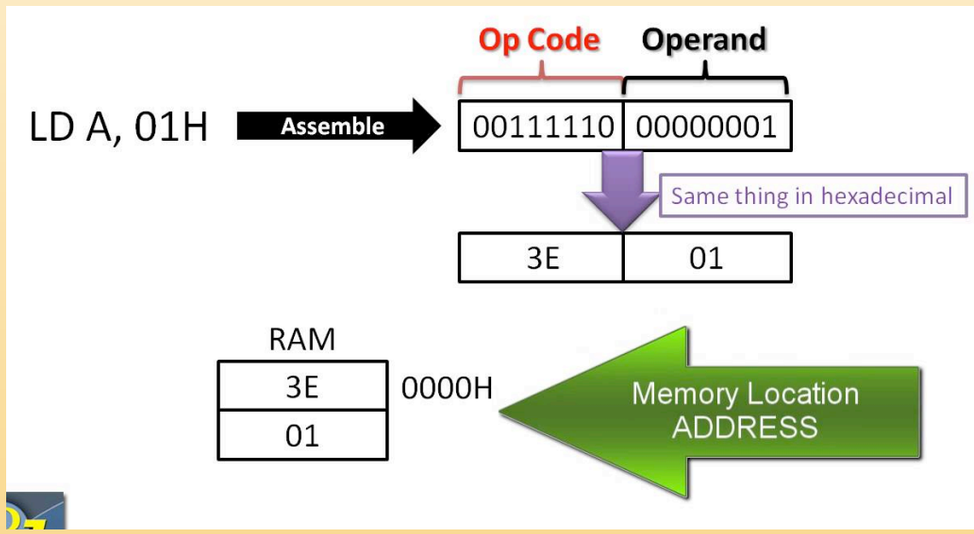
Instrução	Significado
LD B, C	B ← C
LD A, (BC)	A ← Conteúdo apontado pelo endereço formado por BC (16 bits)
LD (HL), A	Guardar no endereço apontado por HL o conteúdo do acumulador A.
ADD A, (HL)	A ← A + Conteúdo apontado pelo endereço indicado por HL (16 bits)
SUB A, B	A ← A – B
XOR C	A ← A ⊕ C
INC r	r ← r + 1
CPL A	A ← /A
JR Z, e	Se Z=0, continue Sw Z=1, PC ← PC + e
SLA r	Shift left A, r bits
SRA r	Shift right A, r bits

Como um programa é guardado na memória:

Endereço do programa

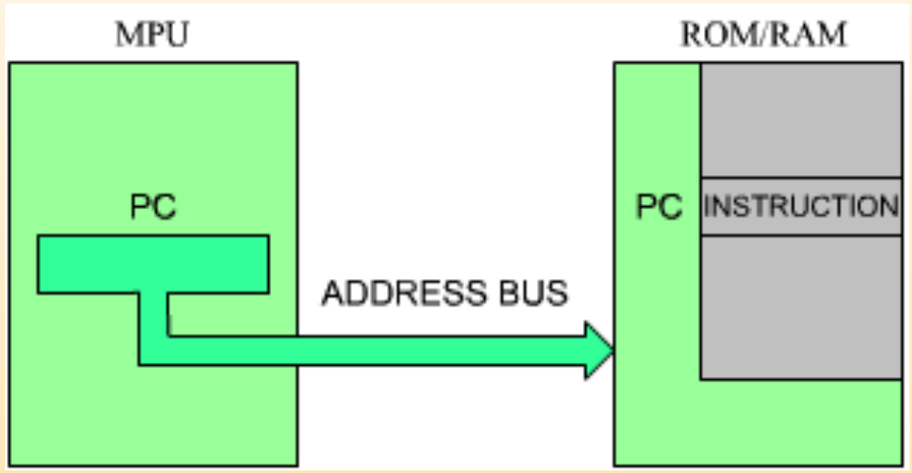
Códigos Hexa

Linguagem Assembly

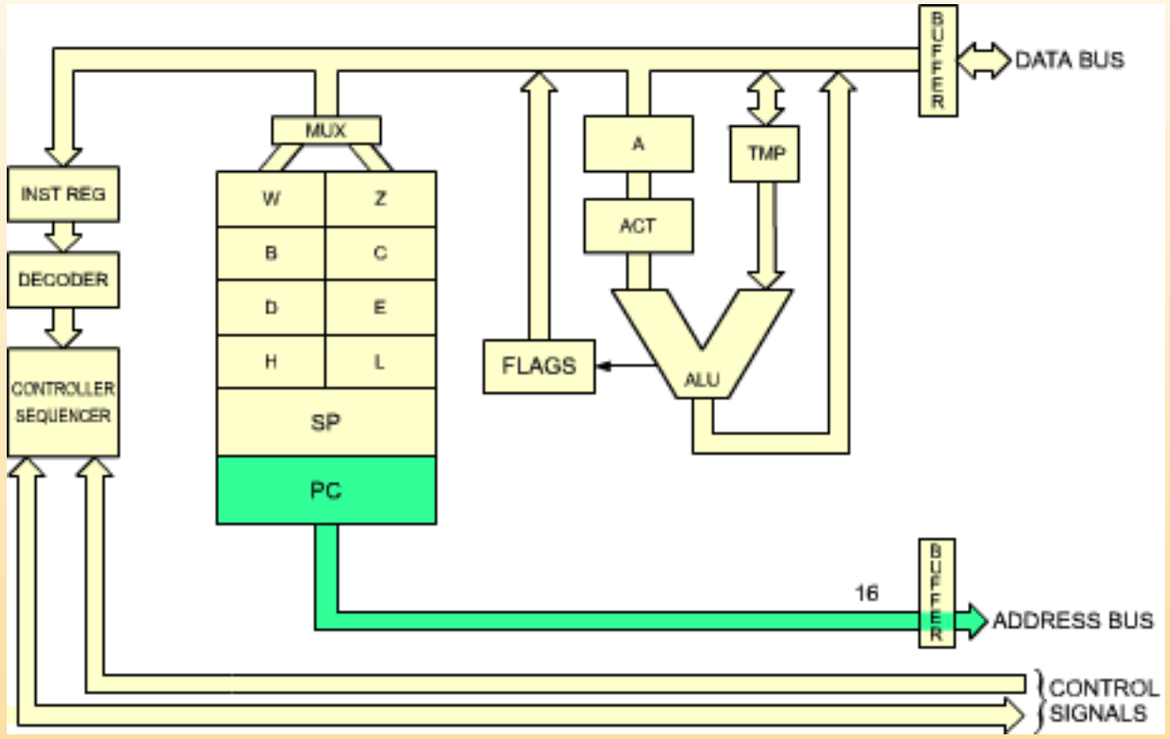


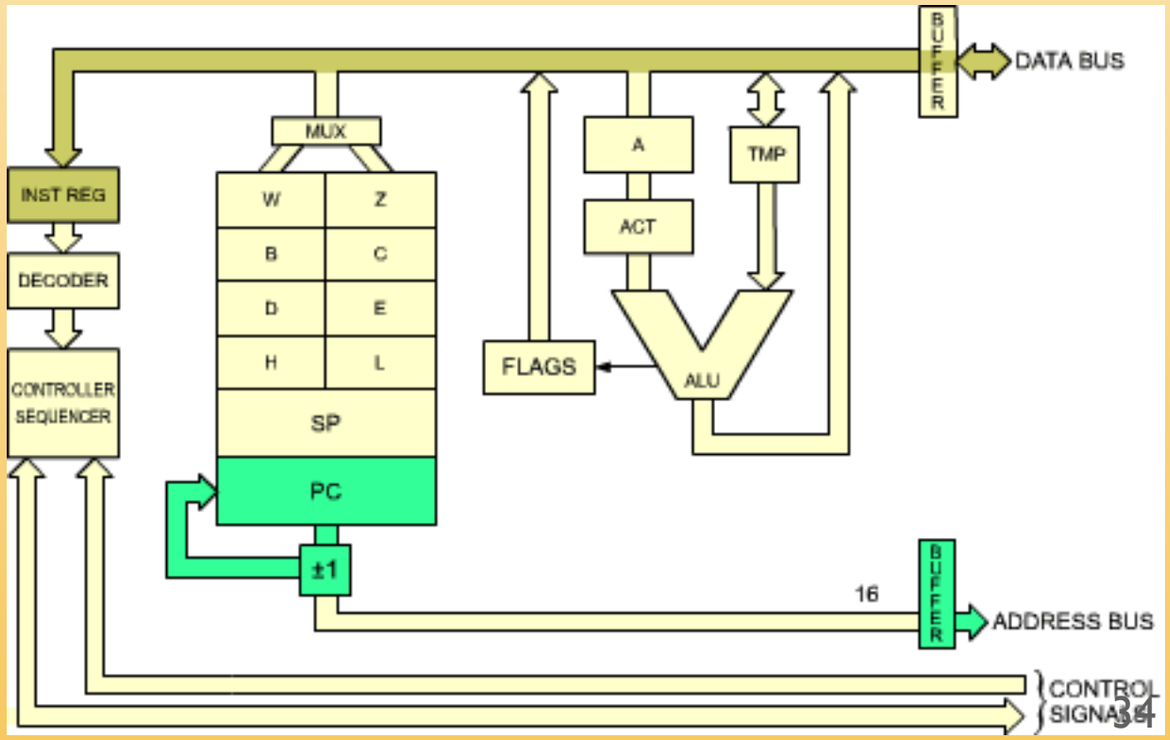

```
0200 A2 EA     LDX SET NO. OF LOOPS FOR 1 SECOND
   2 CA        DEX
   3 A5 60     LDA STORE HOURS IN Fb
   5 85 Fb     STA
   7 A5 61     LDA STORE MIN'S IN FA
   9 85 FA     STA
   b A5 62     LDA STORE SEC'S IN F9
   d 85 F9     STA
   F 86 63     STX SAVE X
  11 84 64     STY (NOT NECESSARY, FILLER)       HR     MIN     SEC
  13 20 1F 1F "SCANDS" (DISPLAY TIME)          1 0     1 0     0 1
  16 A6 63     LDX                              Fb      FA      F9
  18 A4 64     LDY                           (0060)  (0061)  (0062)
  1A E0 00     CPX TO LOOP (TO 0202)
  1C d0 E4     BNE
  1E F8        SED SET DECIMAL MODE TO AVOID HEX DIGITS          -|
  1F 38        SEC SET CARRY                                     |
  20 A9 00     LDA                                               |
  22 65 62     ADC ADD A+C+M-->A (0+1+SEC-->ACC.)                |_ COUNT
  24 85 62     STA STORE IN 62 (SEC) (ACC--> 62)                 |  SECONDS
  26 d8        CLD CLEAR DECIMAL MODE FOR "SCANDS"               |
  27 C9 60     CMP TO LOOP (TO 0200) (RESETTING LOOP FOR NEW SECOND) |
  29 d0 d5     BNE                                               -|
  2b F8        SED                                               -|
  2C 38        SEC SAME AS SECONDS                               |
  2d A9 00     LDA                                               |
  2F 85 62     STA RESET SEC TO 00                               |
  31 65 61     ADC ADD 0+1+MIN-->ACC                             |_ COUNT
  33 85 61     STA STORE IN 61 (MIN) (ACC-->61)                  |  MINUTES
  35 d8        CLD                                               |
  36 C9 60     CMP TO LOOP (TO 0200)                             |
  38 d0 C6     BNE                                               -|
  3A F8        SED SAME AS MINUTES                               -|
  3b 38        SEC                                               |
  3C A9 00     LDA                                               |
  3E 85 62     STA RESET SEC TO 00                               |_ COUNT
  40 85 61     STA RESET MIN TO 00                               |  HOURS
```

- Ciclo de: Busca » Decodificação (da instrução) » Execução:

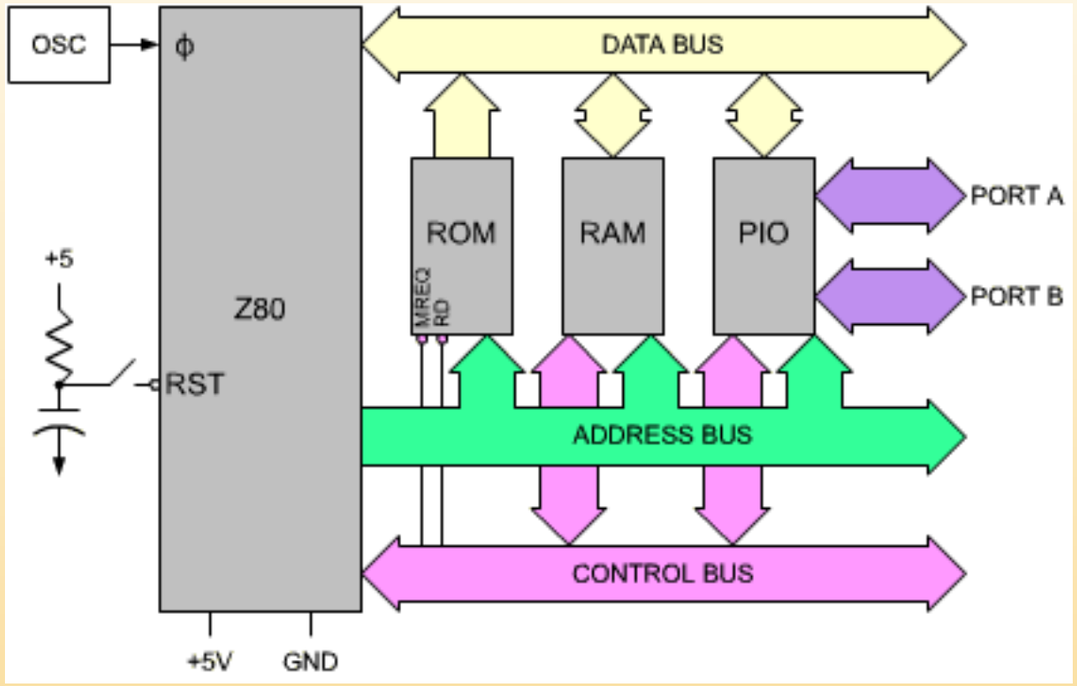


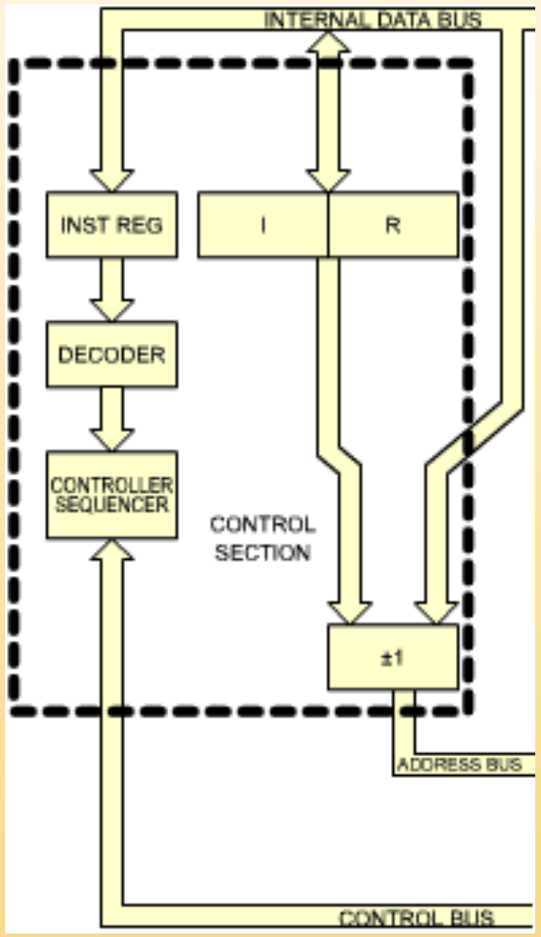
uP busca (lê) no endereço (ou posição de memória) indicado pelo PC (Program Counter) a próxima Instrução à ser executada. O(s) código(s) desta instrução é guardado no IR (Instruction Register). Depois disto, o contador PC é automaticamente incrementado.

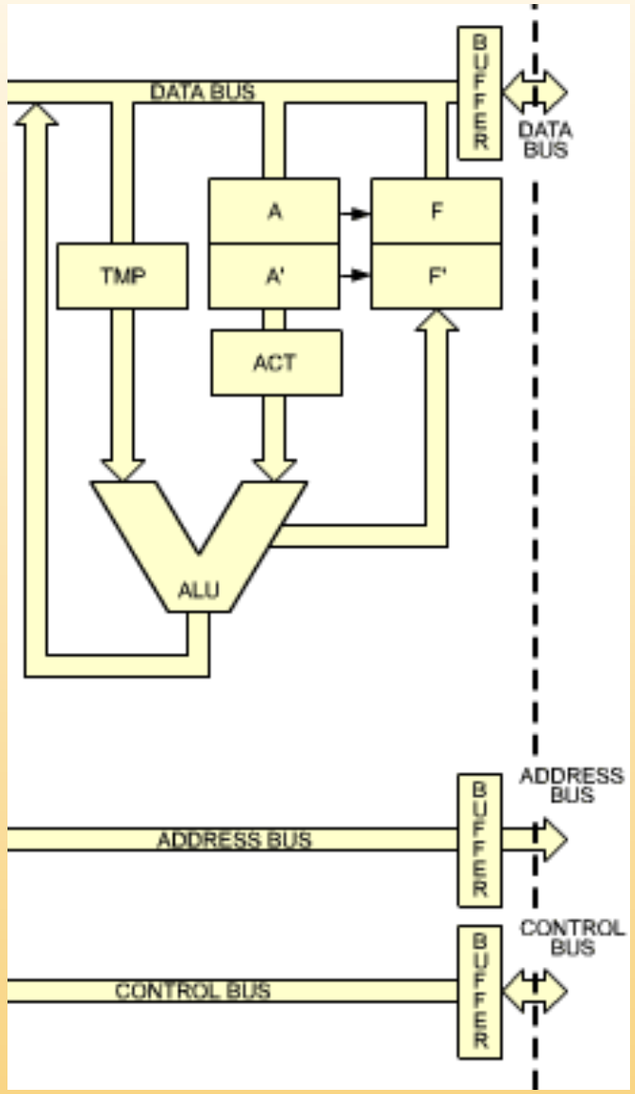




- Ciclo de: Busca » Decodificação (da instrução) » Execução:





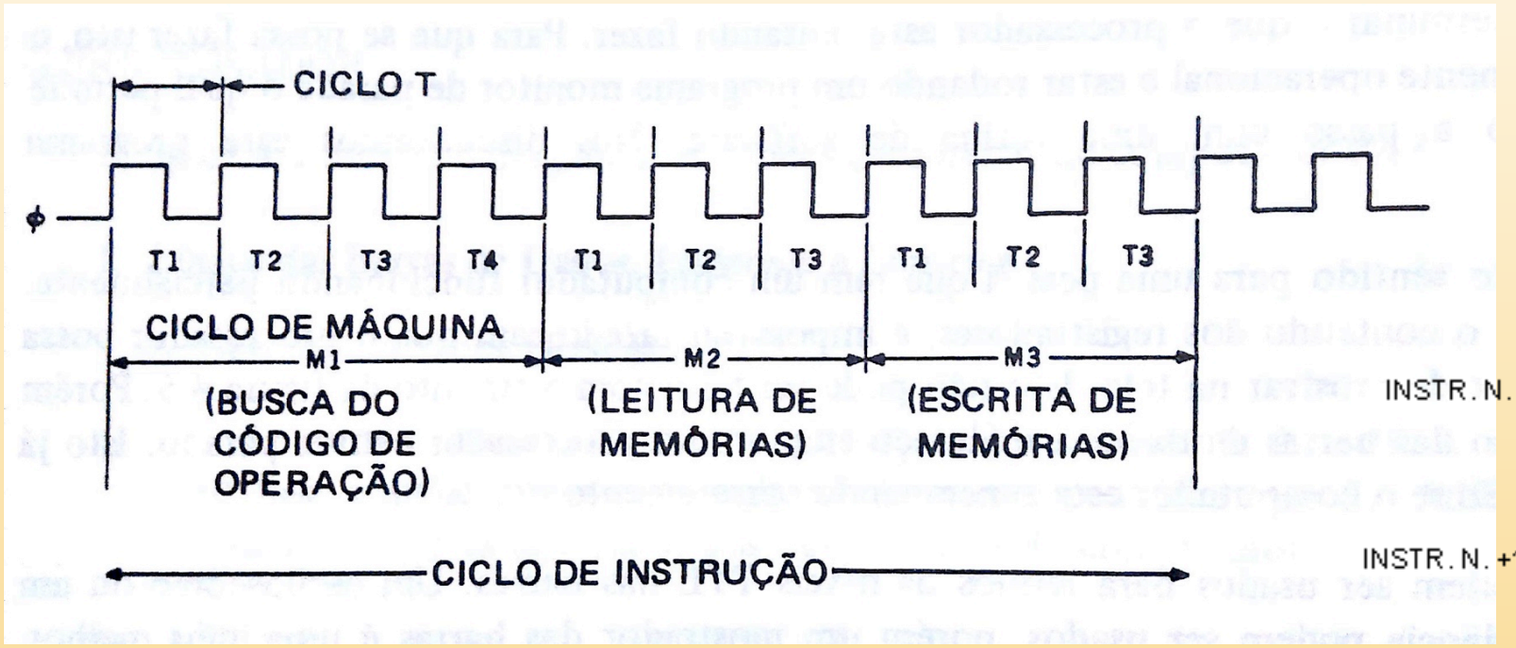


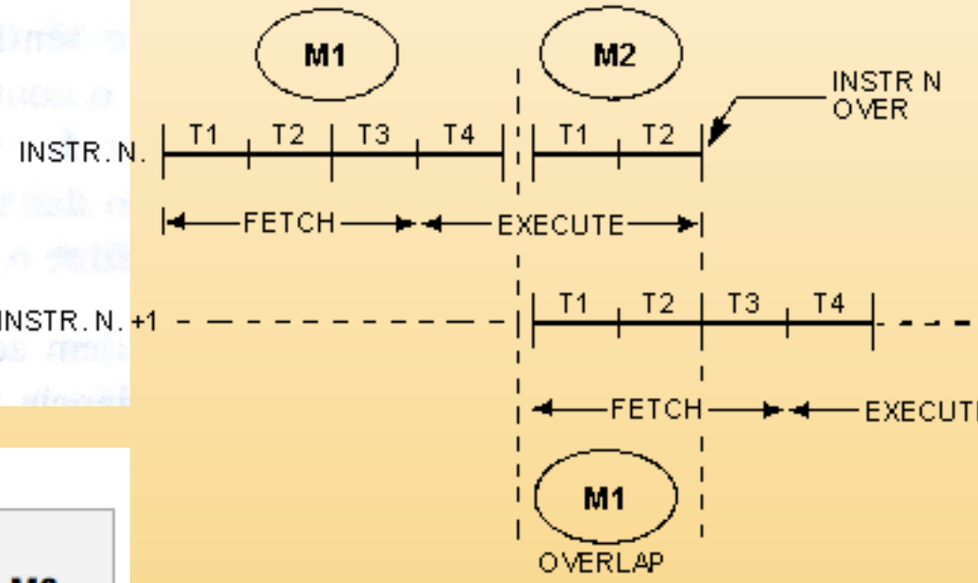
Uma vez que a instrução está contida no IR, a unidade de controle (UC) descodifica o seu conteúdo ("significado") e começa a gerar a sequência correta de sinais internos e externos para lograr a execução da instrução especificada (nos próximos pulsos de clock). Algumas instruções são executadas inteiramente dentro do uP (na sua ULA). Outras instruções dependem de dados externos presentes na memória (e não ainda dentro dos registradores do uP). É por isso que o número de cilcos de clock ou de máquinas varia conforme o tipo de instrução à ser decodificada.

- Exemplo de rotina ASSEMBLY:

```
; memcpy --
; Copy a block of memory from one location to another.
;
; Entry registers
;       BC - Number of bytes to copy
;       DE - Address of source data block
;       HL - Address of target data block
;
; Return registers
;       BC - Zero
1000
1000                    org     1000h       ;Origin at 1000h
1000 78         memcpy  public
1001 B1         loop    ld      a,b         ;Test BC,
1002 C8                 or      c           ;If BC = 0,
1003 1A                 ret     z           ;Return
1004 77                 ld      a,(de)      ;Load A from (DE)
1005 13                 ld      (hl),a      ;Store A into (HL)
1006 23                 inc     de          ;Increment DE
1007 0B                 inc     hl          ;Increment HL
1008 C3 00 10           dec     bc          ;Decrement BC
100B                    jp      loop        ;Repeat the loop
```

- Ciclos de clock (ciclos de máquina) associados com a execução de uma instrução:



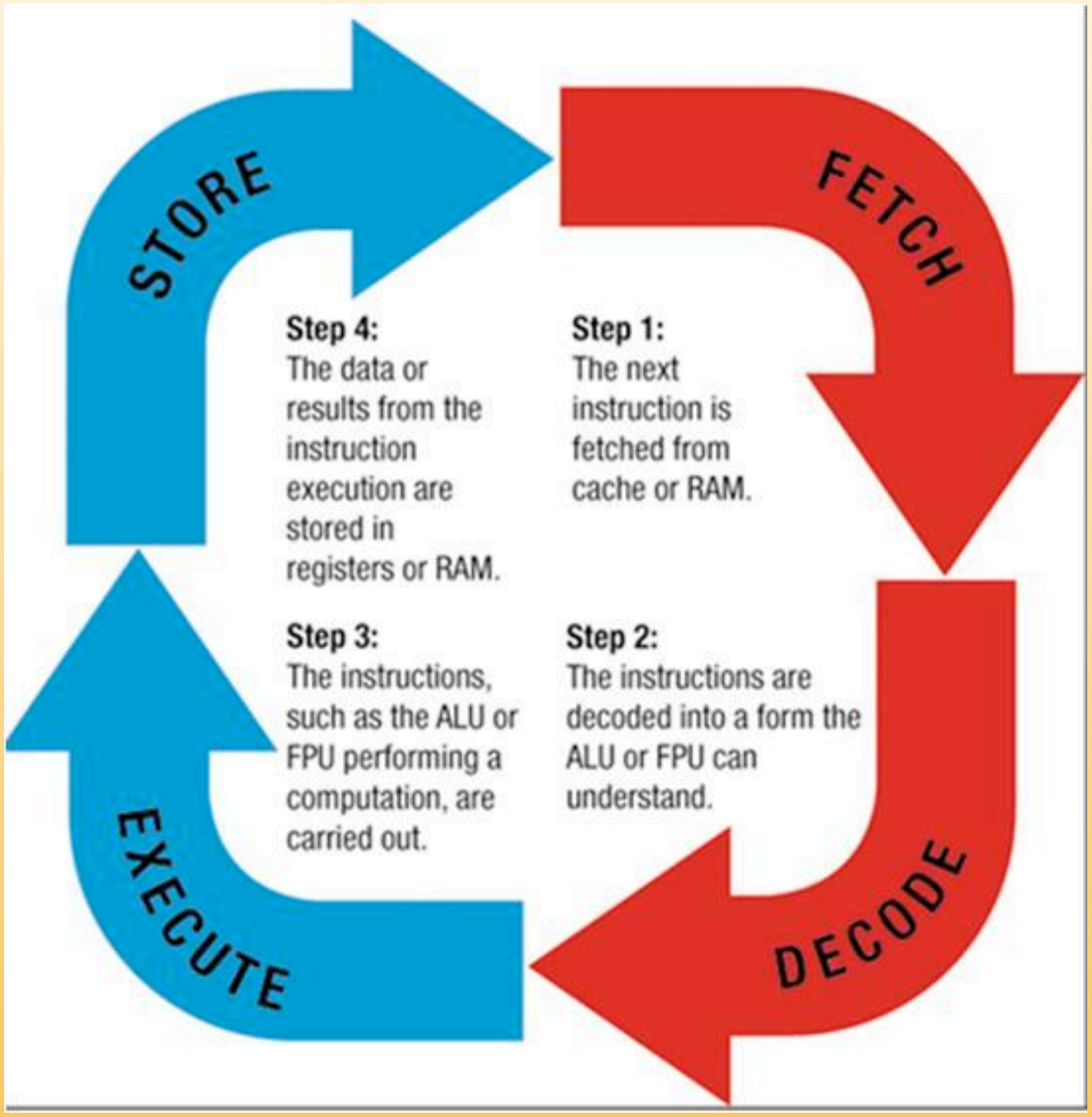


Z80: sobreposição de ciclos de busca...

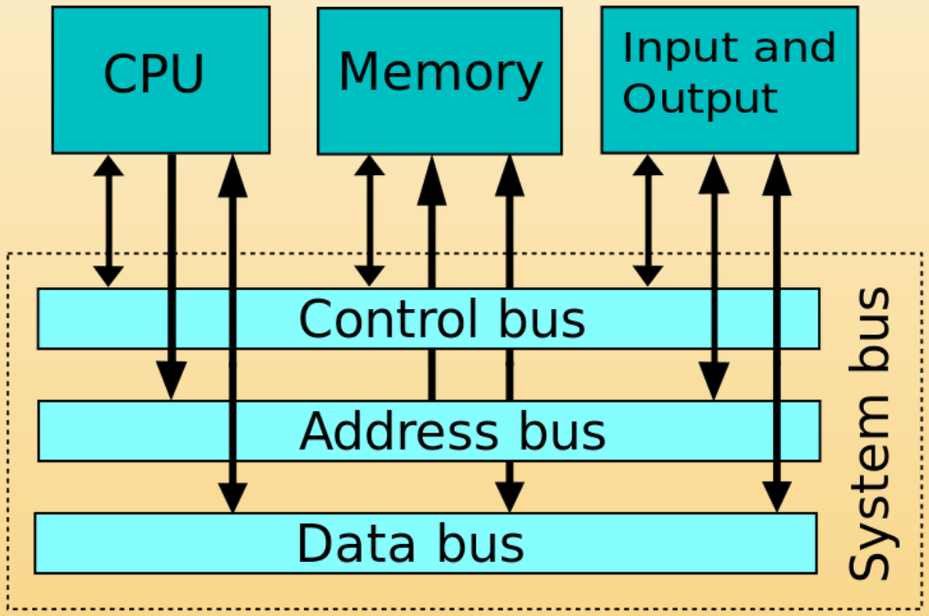
Examples of typical instructions (R=read, W=write)

Total M-cycles	instruction	M1	M2	M3	M4	M5	M6
1[45]	`INC BC`	opcode					
2[46]	`ADD A,n`	opcode	n				
3[47]	`ADD HL,DE`	opcode	internal	internal			
4[48]	`SET b,(HL)`	prefix	opcode	R(HL), set	W(HL)		
5[49]	`LD (IX+d),n`	prefix	opcode	d	n,add	W(IX+d)	
6[50]	`INC (IY+d)`	prefix	opcode	d	add	R(IY+d),inc	W(IY+d)

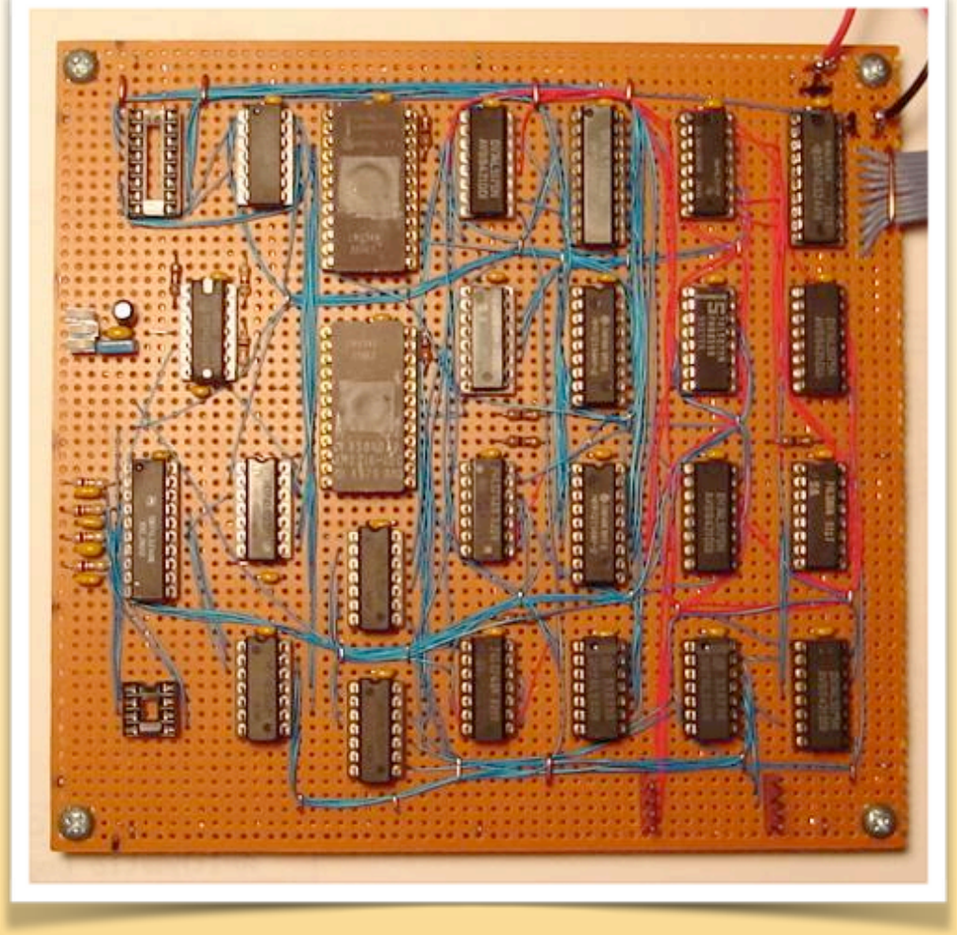
Ciclo de:



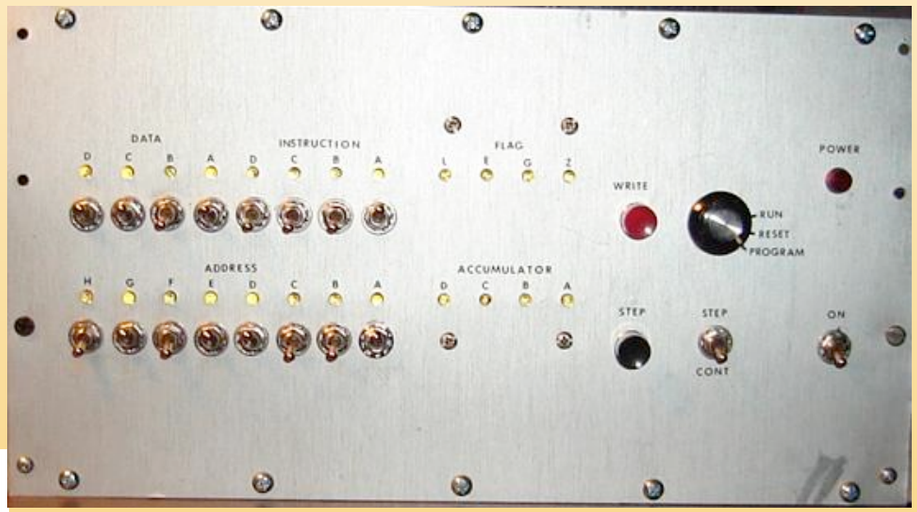
Hardware:



Exemplo uP de 4-bits



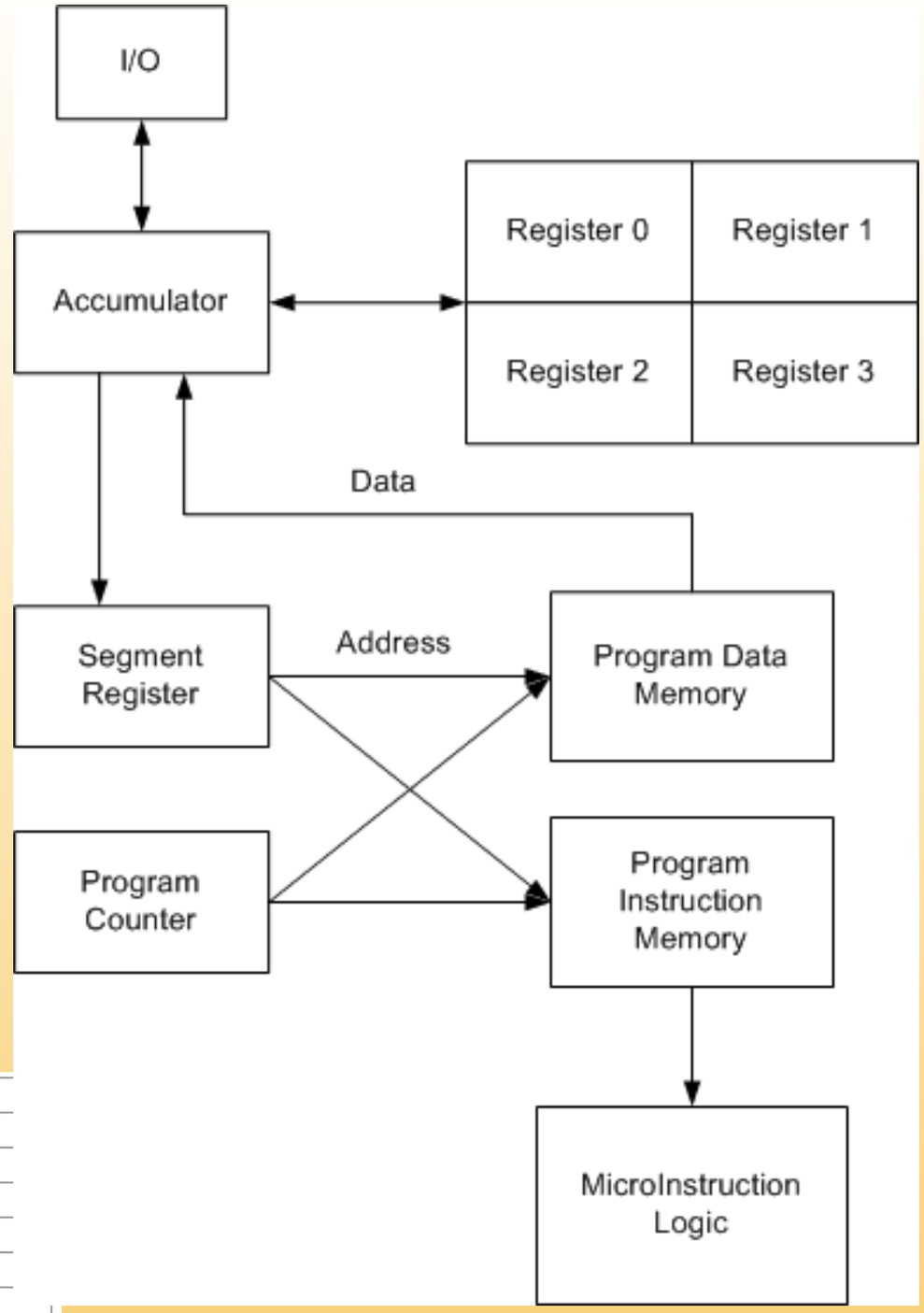
- Página WEB: simples uP de 4-bits: http://www.galacticelectronics.com/Simple4BitCPU.HTML (29.09.2016)



Code	Numonic	Description
1	LIT	Copy a literal value from program memory to the accumulator.
2	LOAD	Copy a value from a register to the accumulator.
3	STORE	Copy the value in the accumulator to a register.
4	INC	Increment the accumulator by one.
5	DEC	Decrement the accumulator by one.
6	REG	Copy the value in the accumulator to the register address latch.
7	CMPL	Compare the accumulator to a literal value. Sets flags.
8	CMPR	Compare the accumulator to a register. Sets flags.
9	RST	Resets the program counter.
10	JUMPL	Copy the value in the accumulator to the segment register when the less than flag is set.
11	JUMPE	Copy the value in the accumulator to the segment register when the equal flag is set.
12	JUMPG	Copy the value in the accumulator to the segment register when the greater than flag is set.
13	IN	Copy the value from the input to the accumulator.
14	OUT	Copy the value in the accumulator to the output latch.
15	NOP	No operation.

Exemplo uP de 4-bits

- Página WEB: simples uP de 4-bits: http://www.galacticelectronics.com/Simple4BitCPU.HTML (29.09.2016)



Code	Numonic	Description
1	LIT	Copy a literal value from program memory to the accumulator.
2	LOAD	Copy a value from a register to the accumulator.
3	STORE	Copy the value in the accumulator to a register.
4	INC	Increment the accumulator by one.
5	DEC	Decrement the accumulator by one.
6	REG	Copy the value in the accumulator to the register address latch.
7	CMPL	Compare the accumulator to a literal value. Sets flags.
8	CMPR	Compare the accumulator to a register. Sets flags.
9	RST	Resets the program counter.
10	JUMPL	Copy the value in the accumulator to the segment register when the less than flag is set.
11	JUMPE	Copy the value in the accumulator to the segment register when the equal flag is set.
12	JUMPG	Copy the value in the accumulator to the segment register when the greater than flag is set.
13	IN	Copy the value from the input to the accumulator.
14	OUT	Copy the value in the accumulator to the output latch.
15	NOP	No operation.

Outras referências:

- Intro to Computer Architecture, vídeo YouTube: https://youtu.be/HEjPop-aK_w (super “básico”).
- The von Neumann architecture, vídeo no YouTube: https://youtu.be/KR2ejxGq464?list=PLCiOXwirraUCaPt5zN4xJTIgKvzVYWa_5
- Fetch Decode Execute Cycle (Immediate Addressing), vídeo no YouTube: https://youtu.be/C6P2s5J9RaM

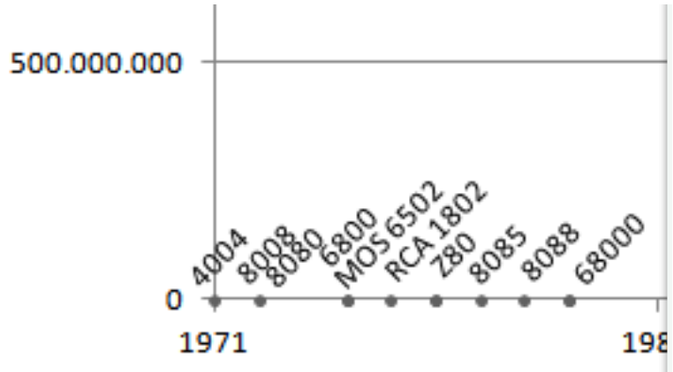
O Almoço Grátis "Acabou"…

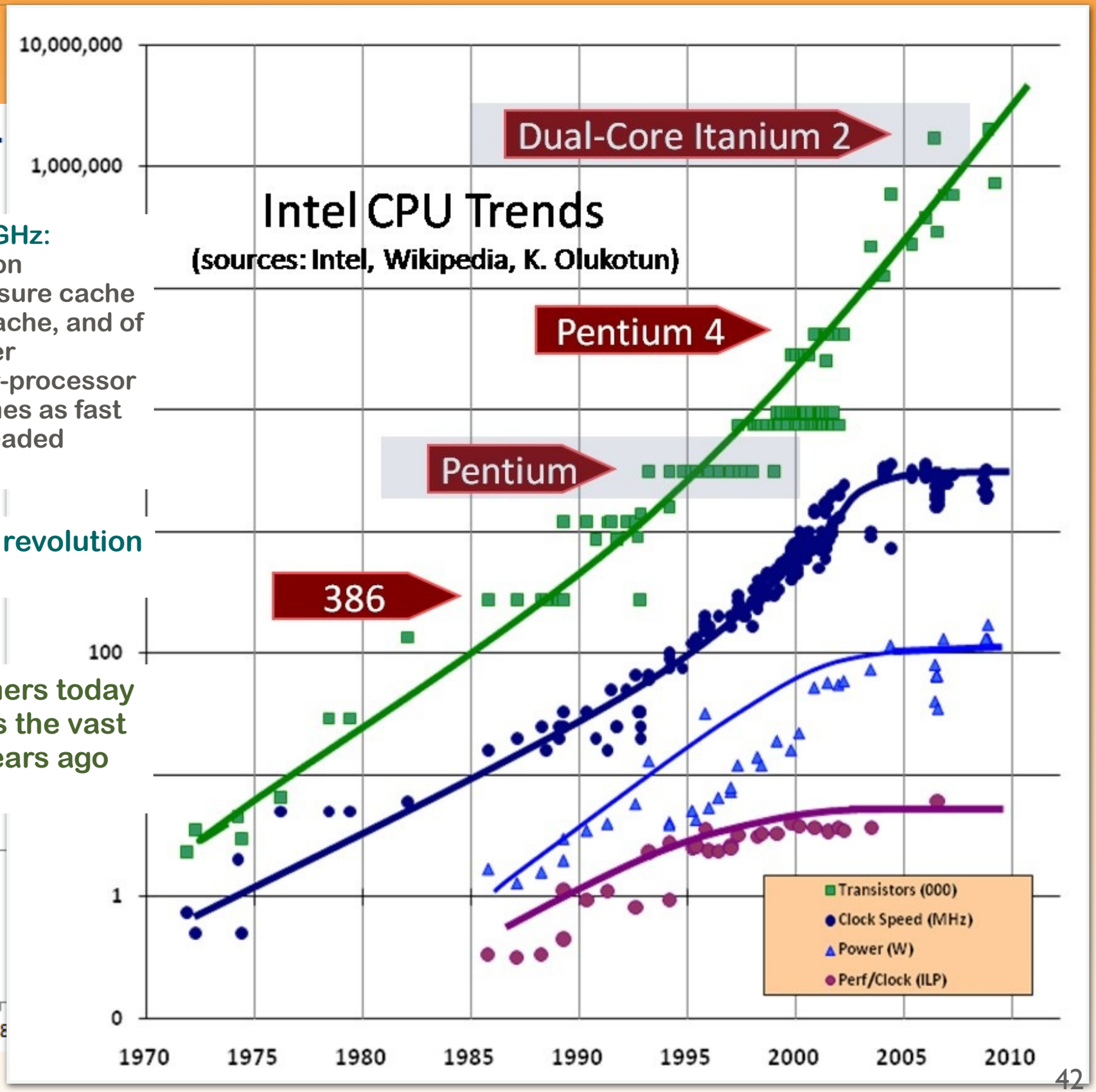
Microprocessor

Myths and Realities: 2 x 3GHz < 6 GHz:
Why not? First, there is coordination overhead between the cores to ensure cache coherency (a consistent view of cache, and of main memory) and to perform other handshaking. Today, a two- or four-processor machine isn't really two or four times as fast as a single CPU even for multi-threaded applications.

Concurrency is the next major revolution in how we write software

The vast majority of programmers today don't grok concurrency, just as the vast majority of programmers 15 years ago didn't yet grok objects.





Artigo de Herb Sutter, Dr. Dobb's Journal, 30 (3), March 2005 (2009).