Computador de 8 bits WR Kits
Set de Instruções

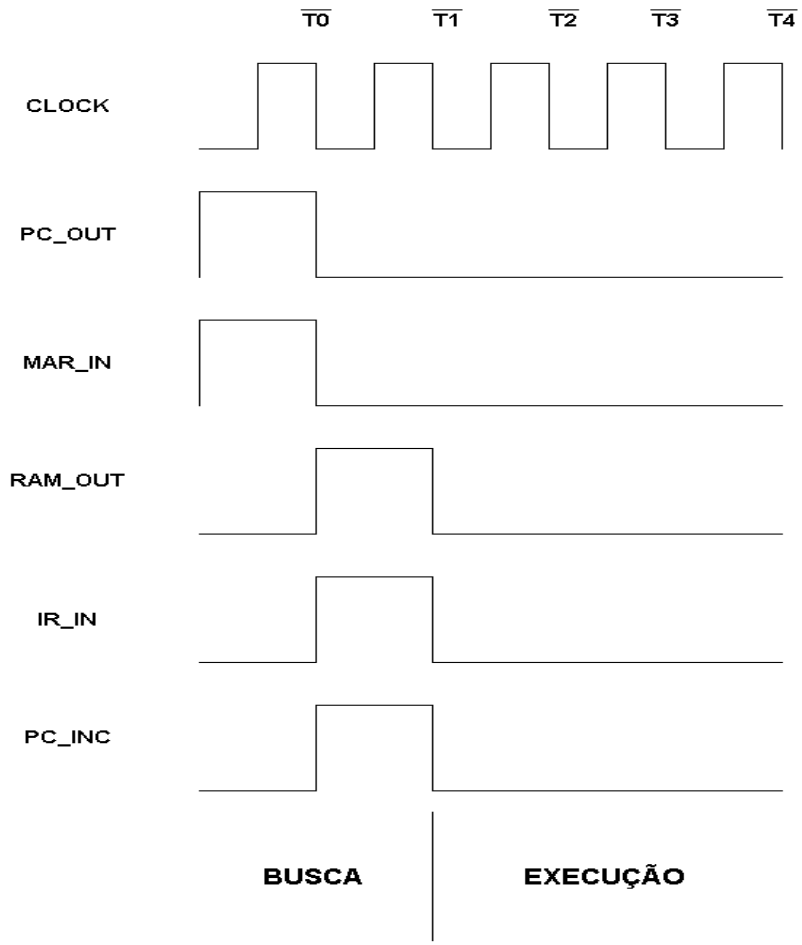
Autor: Eng. Wagner Rambo

Data: Outubro de 2017

www.wrkits.com.br

Ciclo de Busca e Execução

- O ciclo de busca compreende os ciclos de clock $\overline{T0}$ e $\overline{T1}$, sendo a etapa inicial do processamento de qualquer instrução.
- O ciclo de execução muda de instrução para instrução e compreende os ciclos de clock $\overline{T2}$, $\overline{T3}$ e $\overline{T4}$.

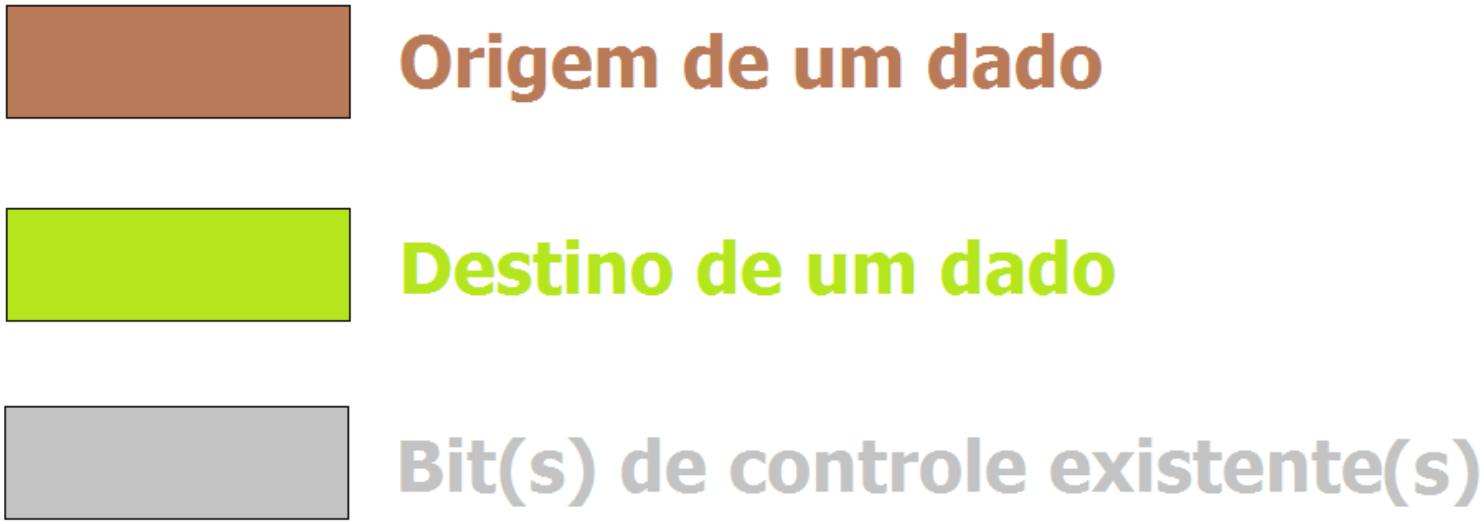


Ciclo de Máquina MC

Tempo em que o processador leva para executar os ciclos de busca e execução de uma única instrução.

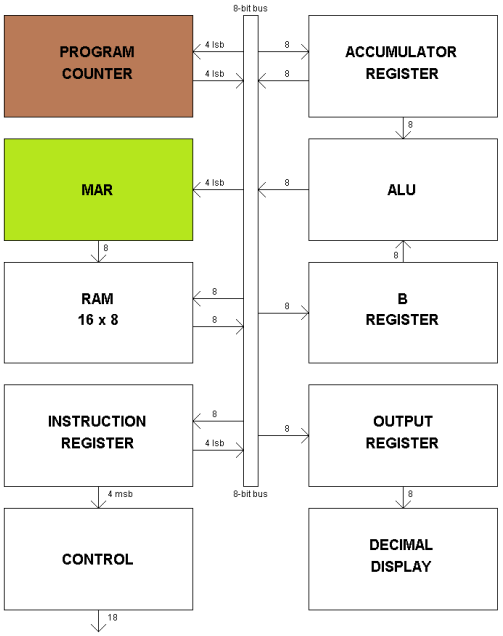
$$MC = \frac{5}{f_{CLOCK}}$$

Legenda de cores dos diagramas de movimentação

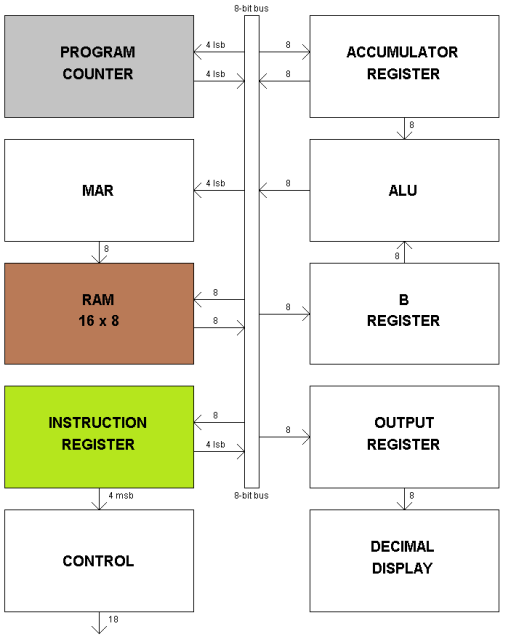


Ciclo de Busca no Computador de 8 bits

$\overline{T0}$



$\overline{T1}$

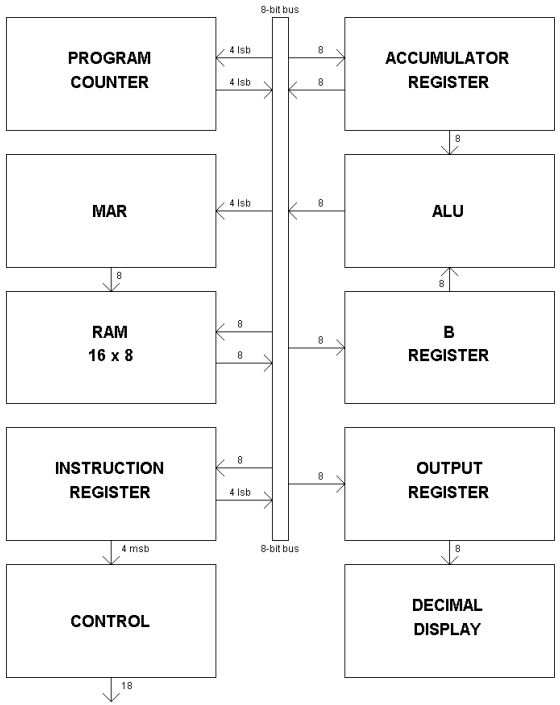


NOP (opcode “0000”)

- No Operation: Sem operação. Processa o ciclo de busca e não realiza qualquer movimentação de dados no ciclo de execução.
- Utilizada para gastar ciclos de máquina, principalmente em rotinas de delay.

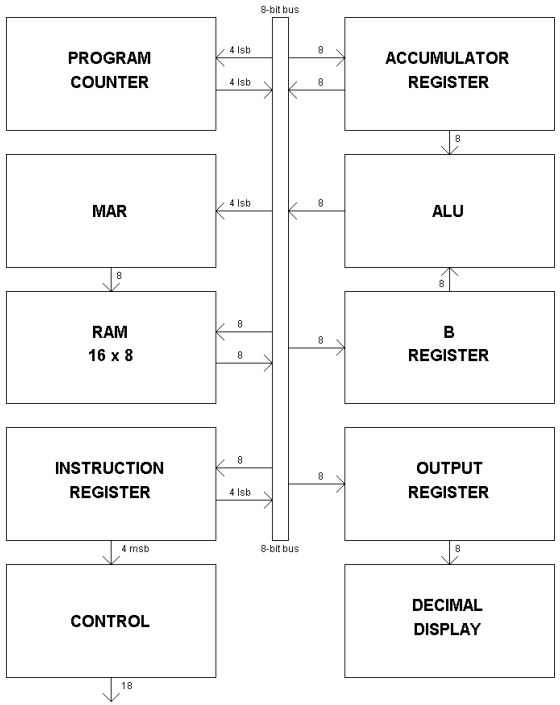
NOP – Ciclo de Execução

$$\overline{T2}$$



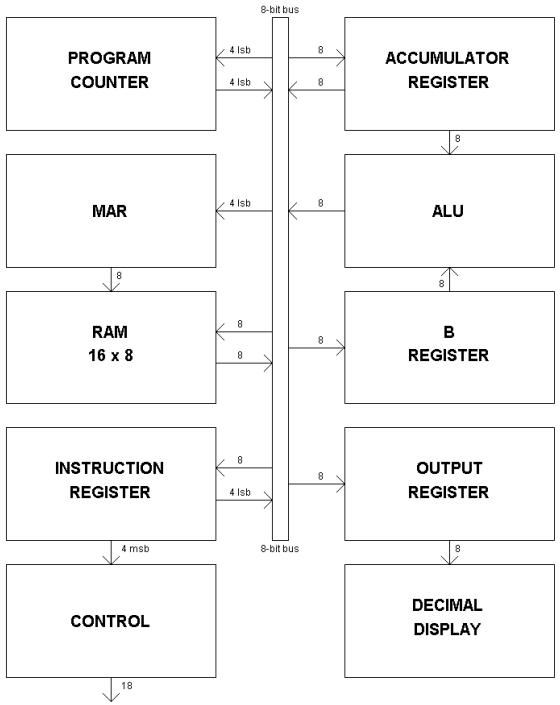
NOP – Ciclo de Execução

$$\overline{T3}$$



NOP – Ciclo de Execução

$$\overline{T4}$$



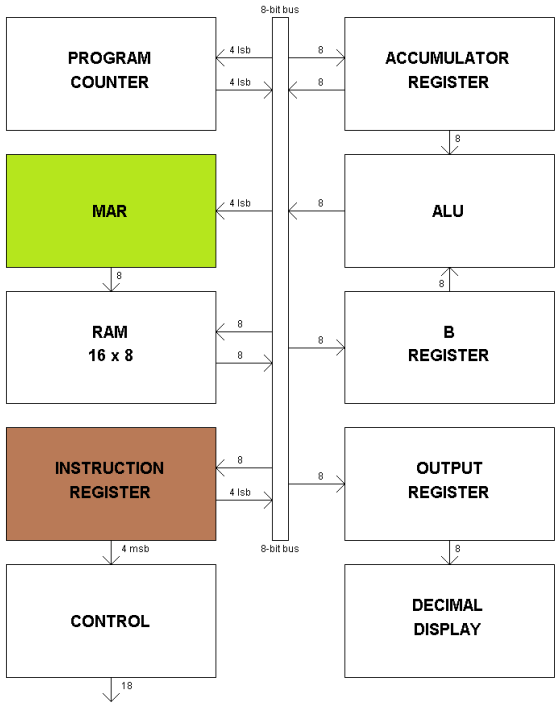
LDA (opcode “0001”)

- Load Accumulator: Carrega acumulador por endereçamento indireto.
- Utilizada para mover um dado de 8 bits para o acumulador. Este dado de 8 bits está contido no endereço apontado pelo operando da instrução (4 bits).
- Exemplo:

```
lda     0Eh     ;move o conteúdo do endereço de memória 0Eh
                ;(1110b) para o acumulador
```

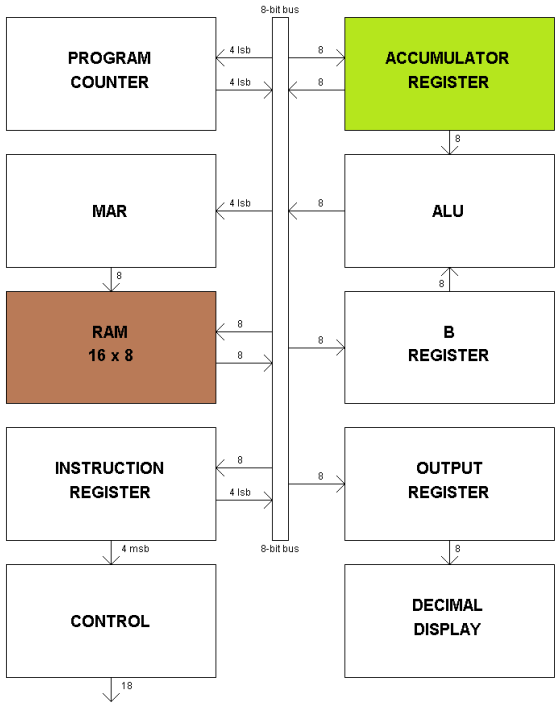
LDA – Ciclo de Execução

$\overline{T2}$



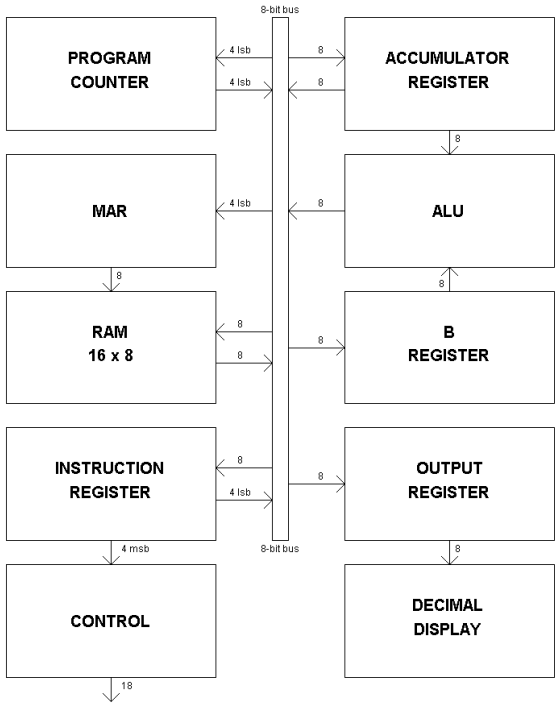
LDA – Ciclo de Execução

$$\overline{T3}$$



LDA – Ciclo de Execução

$\overline{T4}$



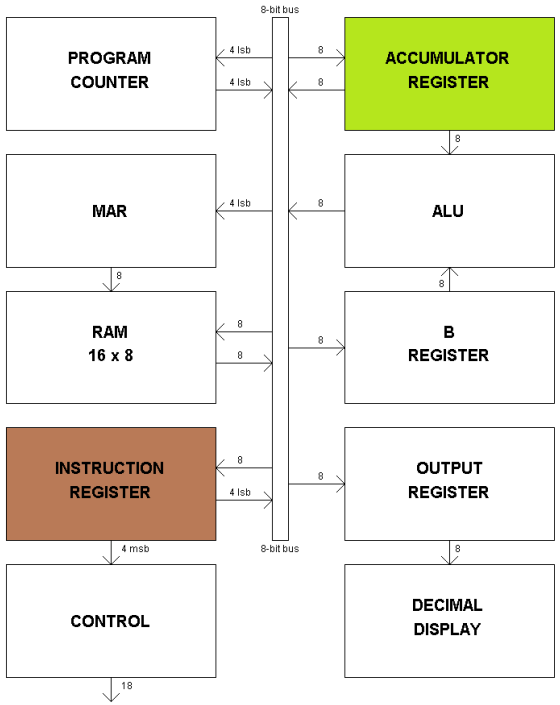
LDI (opcode “0010”)

- Load Immediate: Carrega acumulador por endereçamento imediato.
- Utilizada para mover um dado de no máximo 4 bits para o acumulador. Este dado de 4 bits está contido no operando da instrução (4 bits).
- Exemplo:

```
ldi     09h     ;move o conteúdo 09h
                ;(00001001b) para o acumulador
```

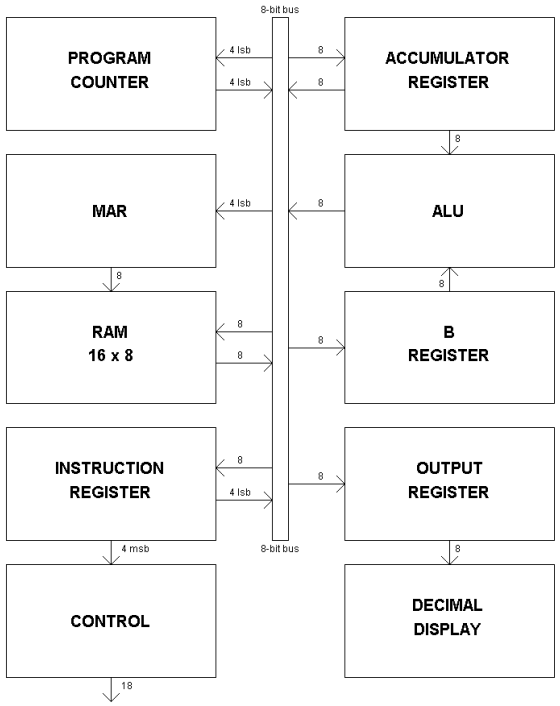
LDI – Ciclo de Execução

$$\overline{T2}$$



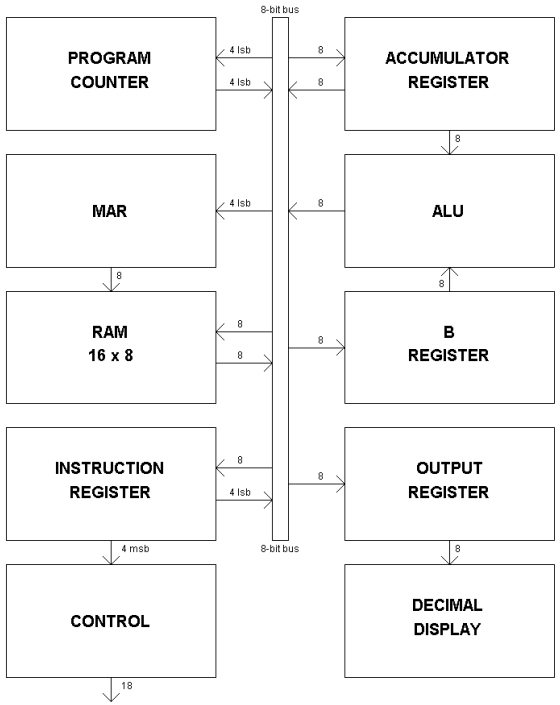
LDI – Ciclo de Execução

$$\overline{T3}$$



LDI – Ciclo de Execução

$\overline{T4}$



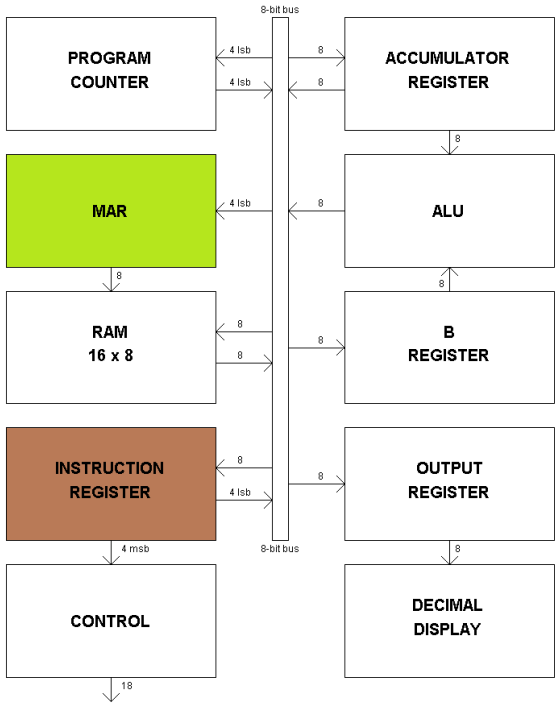
STA (opcode “0011”)

- Store Accumulator: Armazena conteúdo do acumulador em um registrador de uso geral (uma das posições da memória RAM).
- Utilizada para armazenar um dado de 8 bits do acumulador em um endereço de memória.
- Exemplo:

```
sta     0Ch     ;armazena conteúdo do acumulador
                ;no endereço 0Ch (1100b) de memória
```

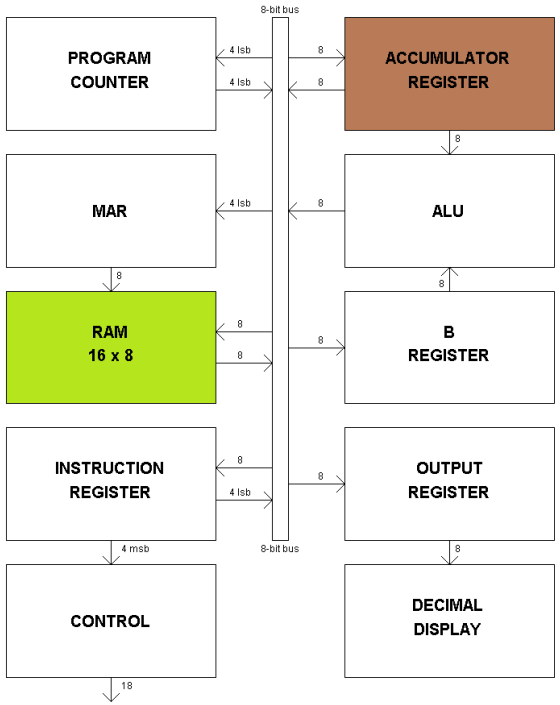
STA – Ciclo de Execução

$$\overline{T2}$$



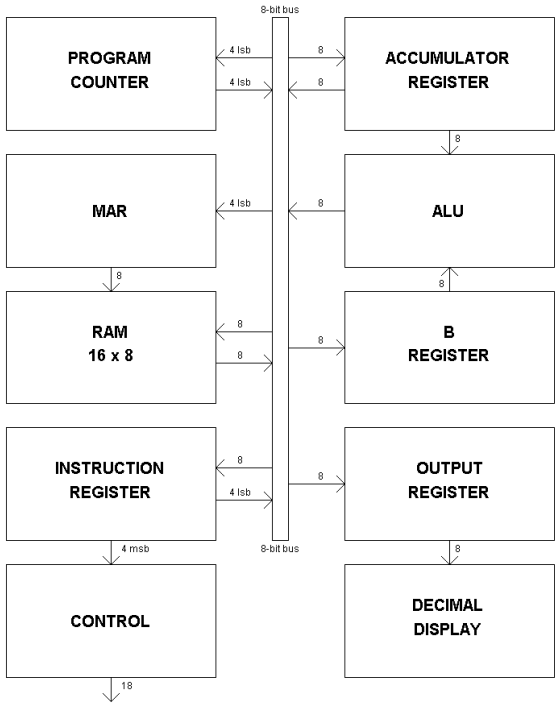
STA – Ciclo de Execução

$$\overline{T3}$$



STA – Ciclo de Execução

$$\overline{T4}$$



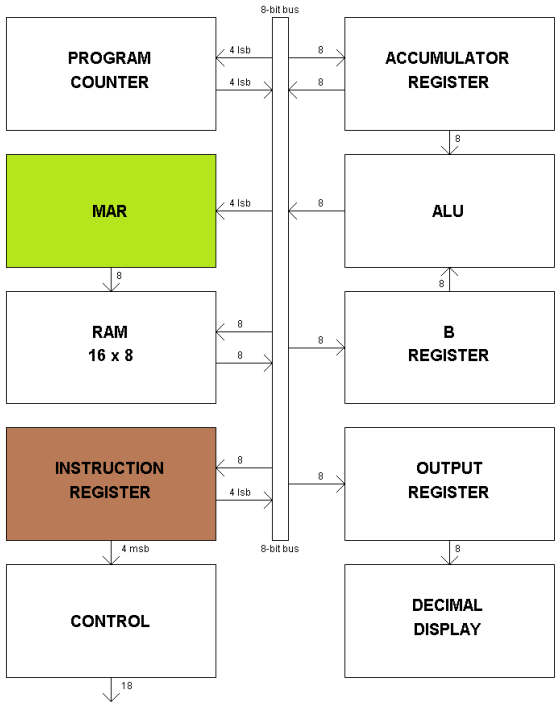
ADD (opcode “0100”)

- Add: Soma o conteúdo do acumulador com o conteúdo da posição de memória apontada pelo operando da instrução. O conteúdo da respectiva posição de memória é salvo no registrador B. O resultado é armazenado no acumulador.
- Utilizada para operação aritmética de soma.
- Exemplo:

```
add    07h    ;soma conteúdo do acumulador com conteúdo
              ;do endereço 07h (0111b) de memória e salva o
              ;resultado no acumulador. ACC = ACC + (07h)
```

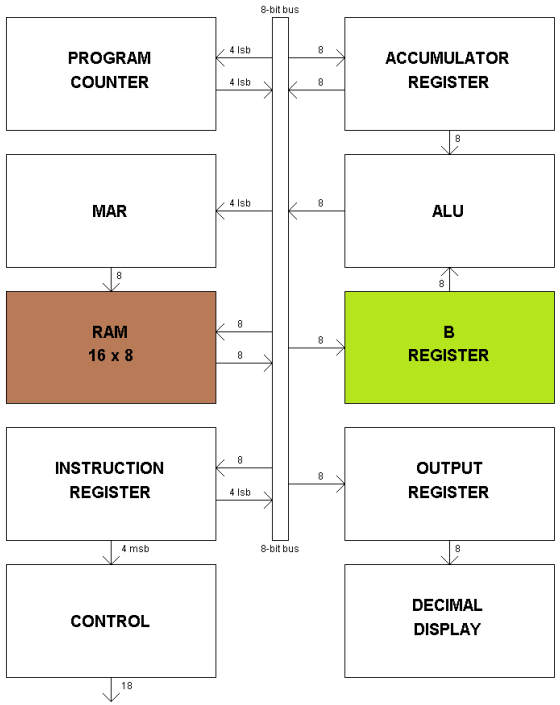
ADD – Ciclo de Execução

$$\overline{T2}$$



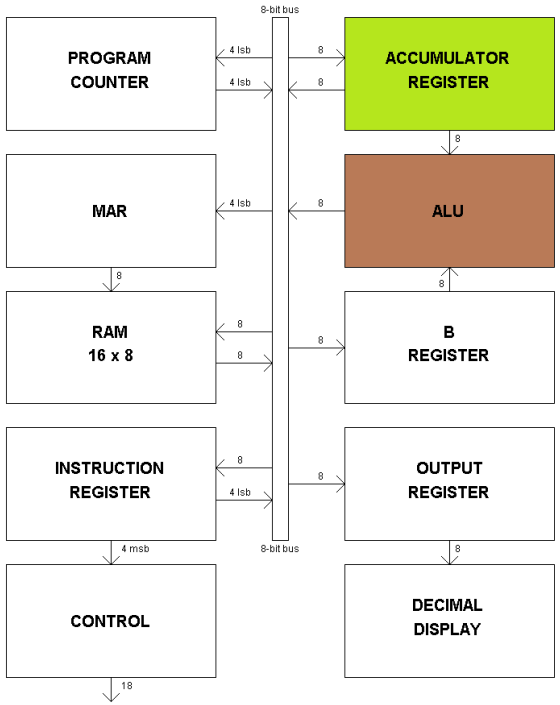
ADD – Ciclo de Execução

$$\overline{T3}$$



ADD – Ciclo de Execução

$$\overline{T4}$$



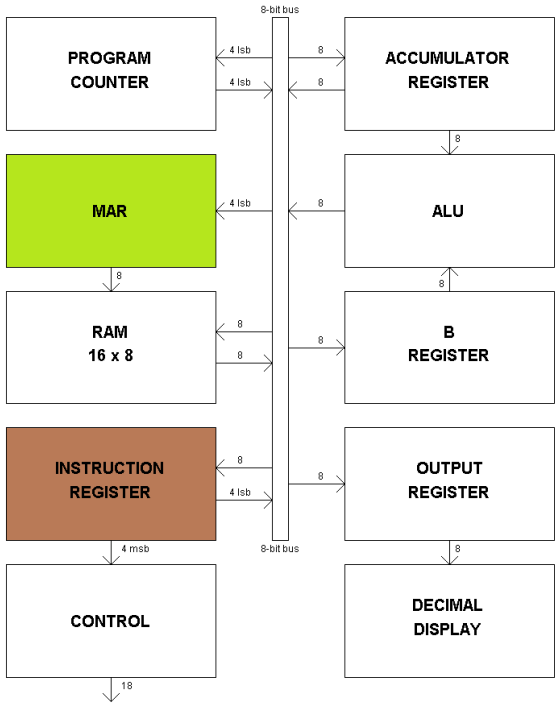
SUB (opcode “0101”)

- Subtract: Subtrai o conteúdo da posição de memória apontada pelo operando da instrução, do acumulador. O conteúdo da respectiva posição de memória é salvo no registrador B. O resultado é armazenado no acumulador.
- Utilizada para operação aritmética de subtração.
- Exemplo:

```
sub    0Dh    ;subtrai conteúdo do endereço 0Dh do acumulador
              ;e salva o resultado no acumulador.
              ;ACC = ACC - (0Dh)
```

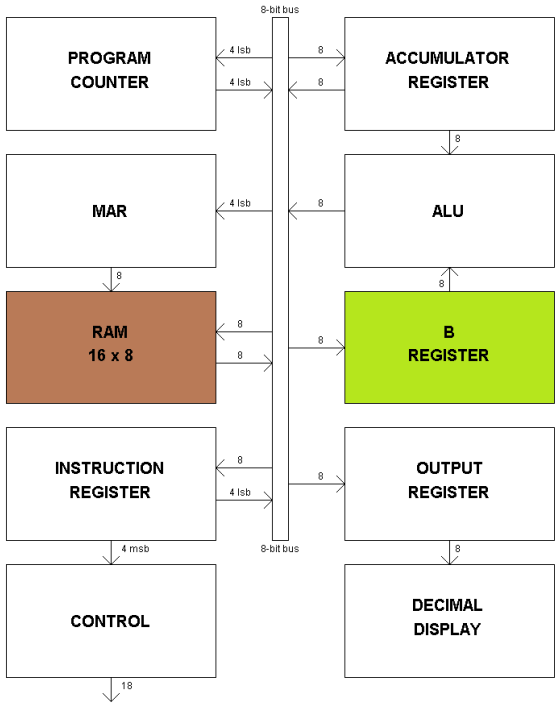
SUB – Ciclo de Execução

$$\overline{T2}$$



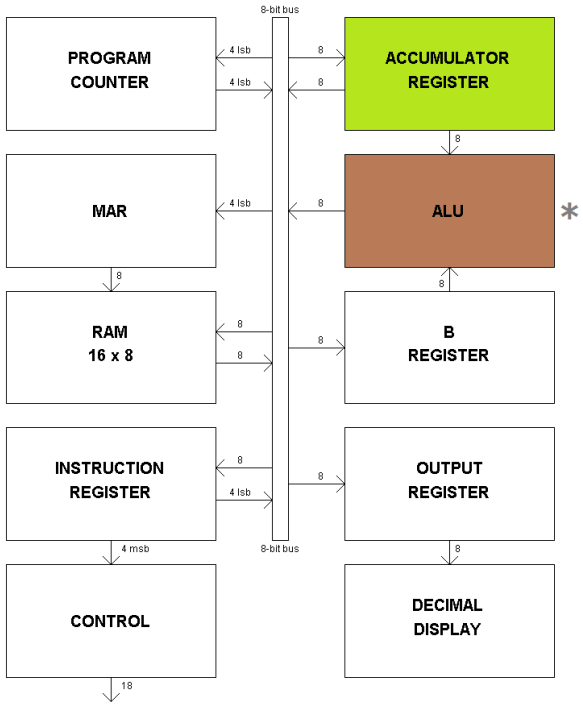
SUB – Ciclo de Execução

$$\overline{T3}$$



SUB – Ciclo de Execução

$\overline{T4}$



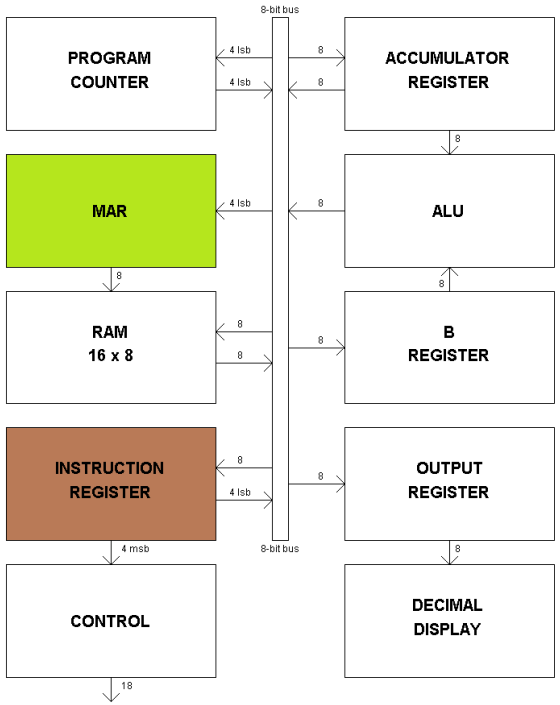
AND (opcode “0110”)

- AND Logic: Realiza a operação lógica AND bit a bit entre o conteúdo do acumulador e o conteúdo da posição de memória apontada pelo operando da instrução. O conteúdo da respectiva posição de memória é salvo no registrador B. Salva o resultado no acumulador.
- Utilizada para operação lógica “E”.
- Exemplo:

```
and    08h    ;operação and entre acumulador e posição de
              ;memória 08h
              ;ACC = ACC AND (08h)
```

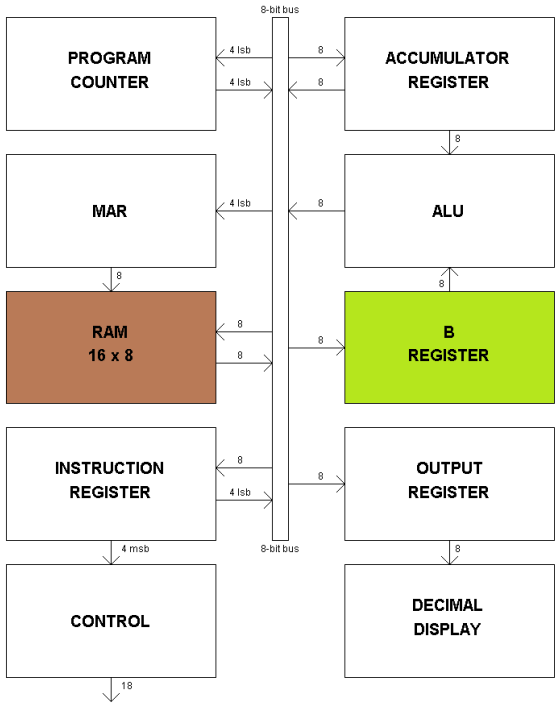
AND – Ciclo de Execução

$$\overline{T2}$$



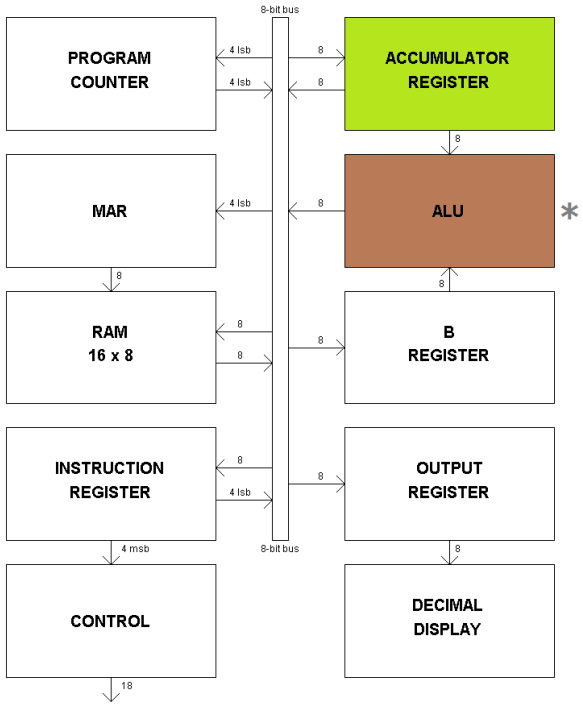
AND – Ciclo de Execução

$$\overline{T3}$$



AND – Ciclo de Execução

$$\overline{T4}$$



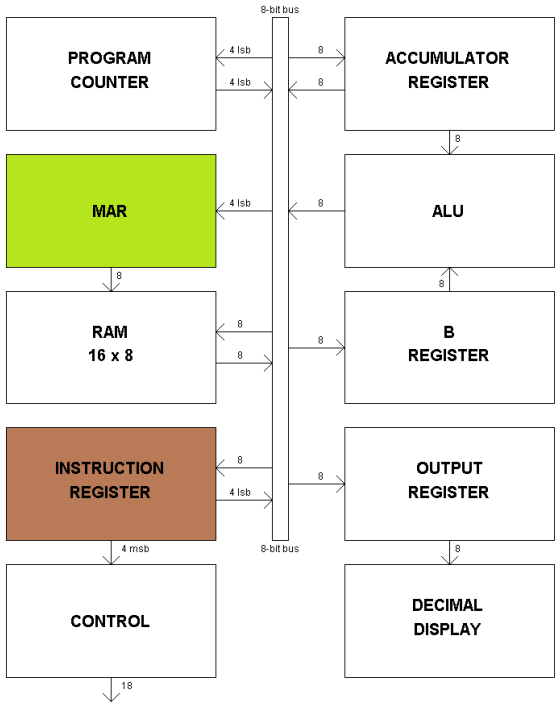
ORL (opcode “0111”)

- OR Logic: Realiza a operação lógica OR bit a bit entre o conteúdo do acumulador e o conteúdo da posição de memória apontada pelo operando da instrução. O conteúdo da respectiva posição de memória é salvo no registrador B. Salva o resultado no acumulador.
- Utilizada para operação lógica “OU”.
- Exemplo:

```
orl     09h     ;operação or entre acumulador e posição de
                ;memória 09h
                ;ACC = ACC OR (09h)
```

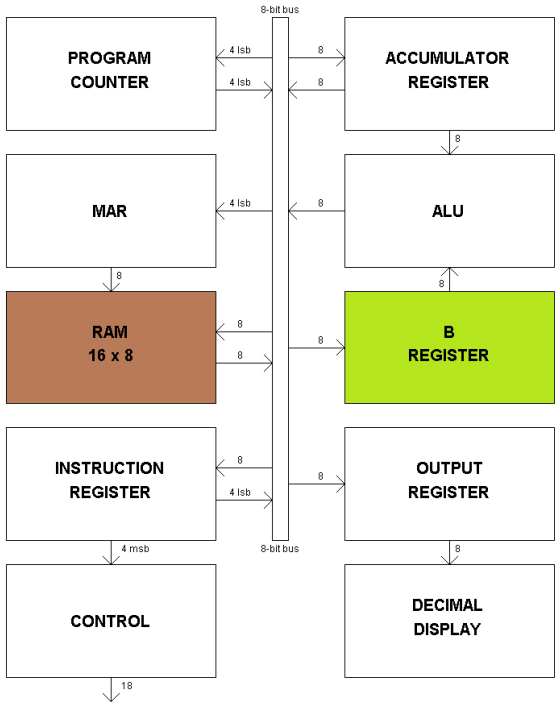
ORL – Ciclo de Execução

$\overline{T2}$



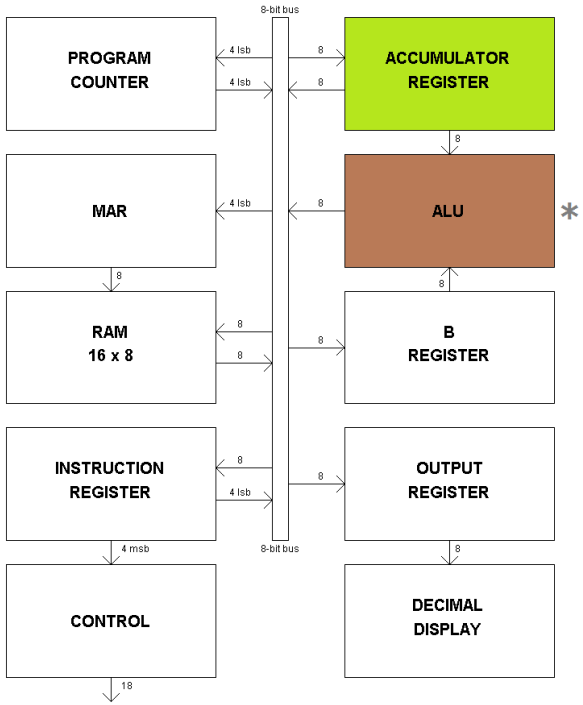
ORL – Ciclo de Execução

$$\overline{T3}$$



ORL – Ciclo de Execução

$\overline{T4}$



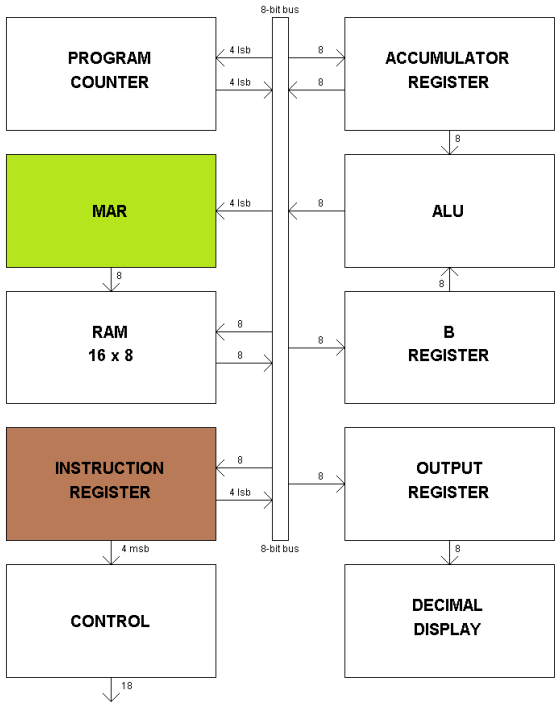
XOR (opcode “1000”)

- XOR Logic: Realiza a operação lógica XOR bit a bit entre o conteúdo do acumulador e o conteúdo da posição de memória apontada pelo operando da instrução. O conteúdo da respectiva posição de memória é salvo no registrador B. Salva o resultado no acumulador.
- Utilizada para operação lógica “OU-Exclusivo”.
- Exemplo:

```
xor     0Ah     ;operação xor entre acumulador e posição de
                ;memória 0Ah
                ;ACC = ACC XOR (0Ah)
```

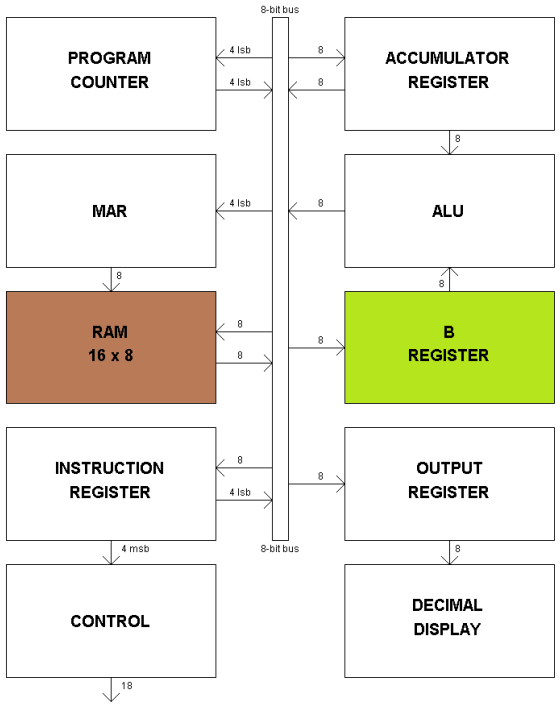
XOR – Ciclo de Execução

$$\overline{T2}$$



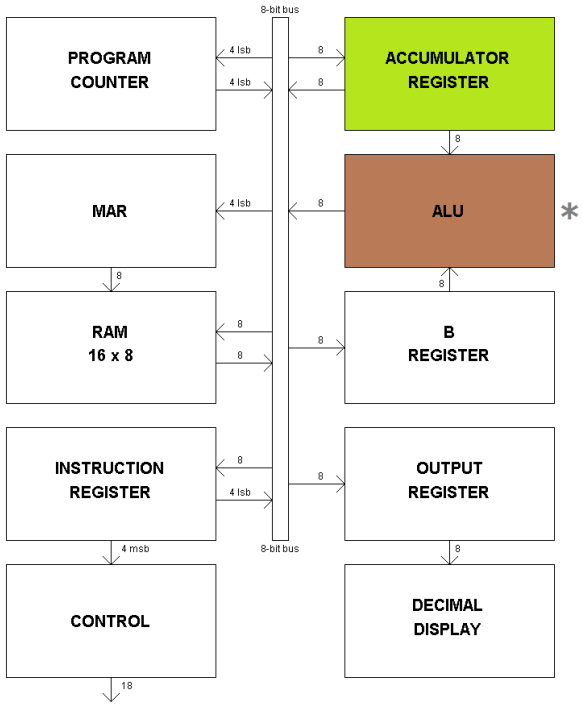
XOR – Ciclo de Execução

$$\overline{T3}$$



XOR – Ciclo de Execução

$$\overline{T4}$$



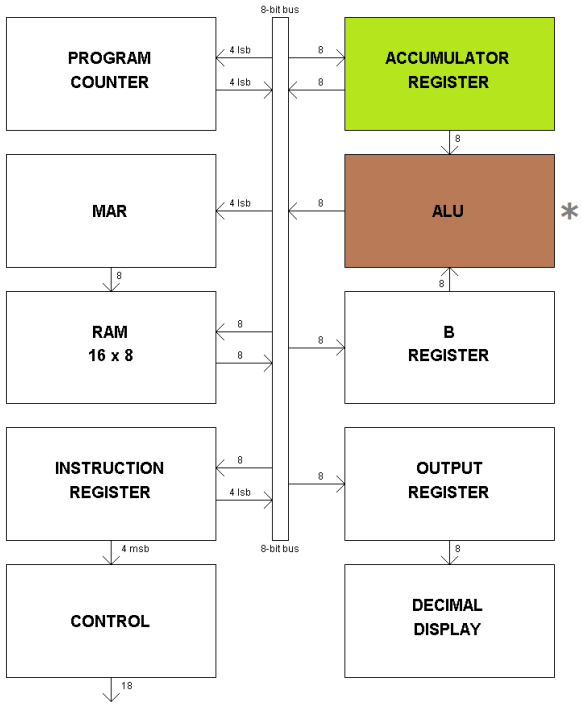
NOT (opcode “1001”)

- NOT: Realiza a operação lógica NOT bit a bit no acumulador. Salva o resultado no próprio acumulador.
- Utilizada para operação lógica de complemento de um.
- Exemplo:

```
not             ;operação not no acumulador
                ;ACC = NOT ACC
```

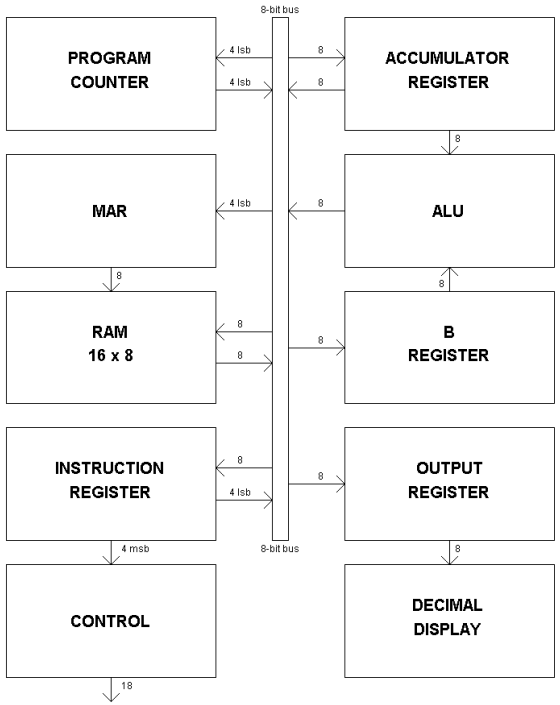
NOT – Ciclo de Execução

$$\overline{T2}$$



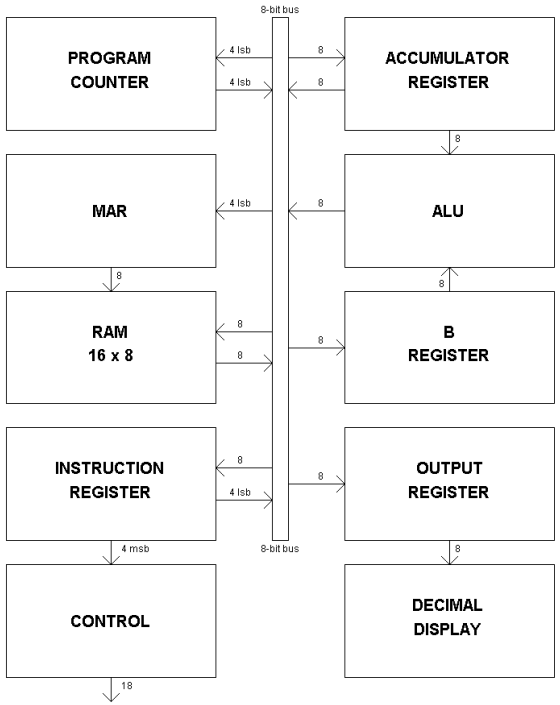
NOT – Ciclo de Execução

$$\overline{T3}$$



NOT – Ciclo de Execução

$$\overline{T4}$$



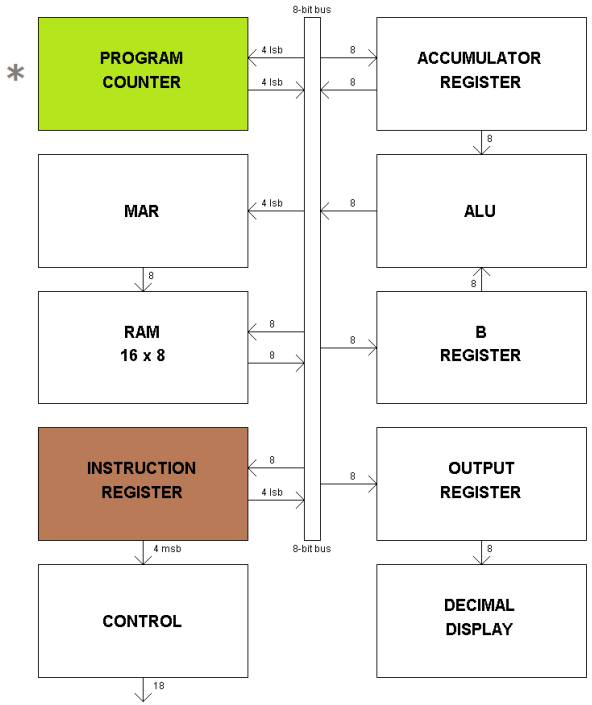
JMP (opcode “1010”)

- Jump: Realiza um desvio incondicional para o endereço de memória apontado pelo operando da instrução.
- Utilizada para realização de loops no programa.
- Exemplo:

```
jmp    02h    ;desvio incondicional para o endereço relativo
              ;02h (0010b) de memória.
```

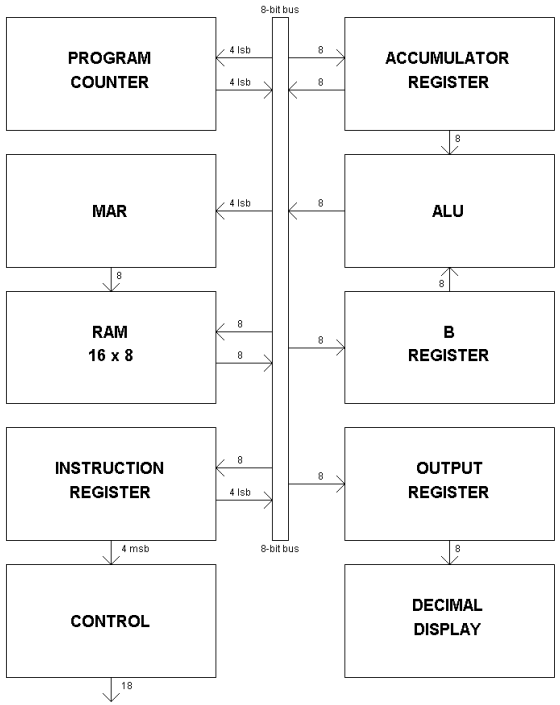
JMP – Ciclo de Execução

$$\overline{T2}$$



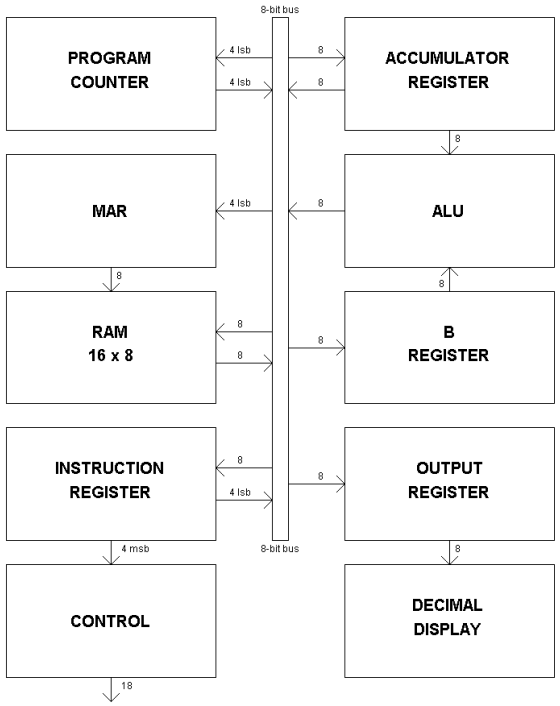
JMP – Ciclo de Execução

$$\overline{T3}$$



JMP – Ciclo de Execução

$$\overline{T4}$$



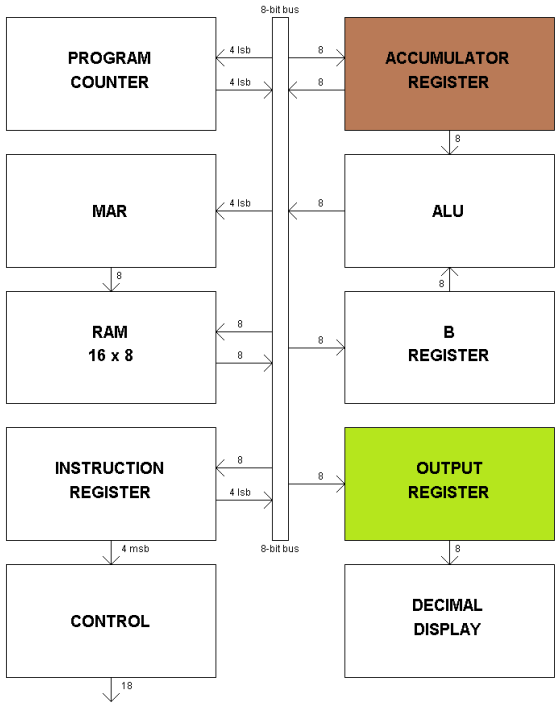
OUT (opcode “1110”)

- Output: Envia o conteúdo do acumulador para o registrador de saída.
- Utilizada para visualização dos resultados de cálculos e programas desenvolvidos no computador de 8 bits.
- Exemplo:

```
out         ;envia o conteúdo do acumulador para
            ;o registrador de saída.
```

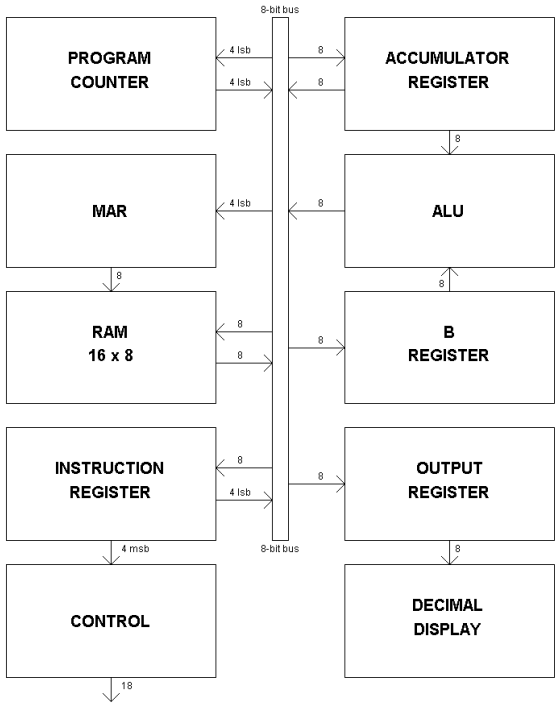
OUT – Ciclo de Execução

$$\overline{T2}$$



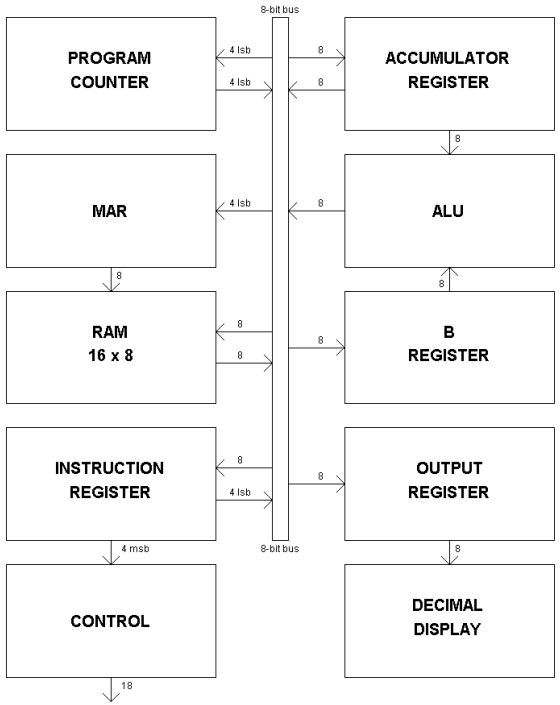
OUT – Ciclo de Execução

$$\overline{T3}$$



OUT – Ciclo de Execução

$$\overline{T4}$$



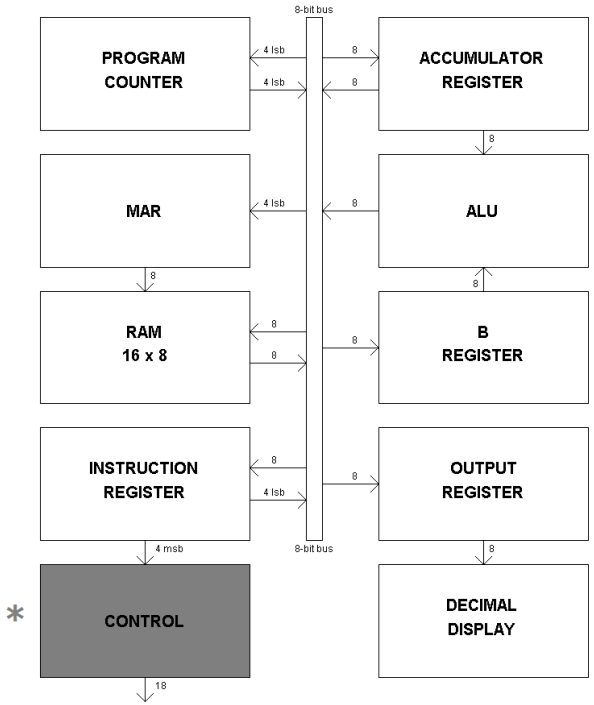
HLT (opcode “1111”)

- Halt: Para o processamento pelo desligamento do sinal de clock.
- Utilizada para paralisar o processamento, comumente quando deseja-se visualizar o resultado de alguma operação do computador no display e/ou registrador de saída.
- Exemplo:

```
hlt             ;para o processamento
                ;
```

HLT – Ciclo de Execução

$$\overline{T2}$$



Resumo do Set de Instruções

Mnemônico	Operando	OPCODE	$\overline{T2}$	$\overline{T3}$	$\overline{T4}$	Operação
NOP	-	0000	-	-	-	-
LDA	End. Mem.	0001	IRO,MAI	RMO,ACI	-	ACC = (End. Mem.)
LDI	Cont. 4 bits	0010	IRO, ACI	-	-	ACC = Operando
STA	End. Mem.	0011	IRO, MAI	ACO, RMI	-	End. Mem. = ACC
ADD	End. Mem.	0100	IRO, MAI	RMO, BRI	ALU, ACI	ACC = ACC + (End. Mem.)
SUB	End. Mem.	0101	IRO, MAI	RMO, BRI	ALU, ACI, SUB	ACC = ACC - (End. Mem.)
AND	End. Mem.	0110	IRO, MAI	RMO, BRI	ALU, ACI, AL0	ACC = ACC AND (End. Mem.)
ORL	End. Mem.	0111	IRO, MAI	RMO, BRI	ALU, ACI, AL1	ACC = ACC OR (End. Mem.)
XOR	End. Mem.	1000	IRO, MAI	RMO, BRI	ALU, ACI, AL1, AL0	ACC = ACC XOR (End. Mem.)
NOT	-	1001	ALU, ACI, AL1, AL0, NOT	-	-	ACC = NOT ACC
JMP	End. Mem.	1010	IRO, JMP	-	-	PC = Operando
OUT	-	1110	ACO, ORI	-	-	Output = ACC
HLT	-	1111	HLT	-	-	Para o sinal de clock