INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

LEIC, LEETC

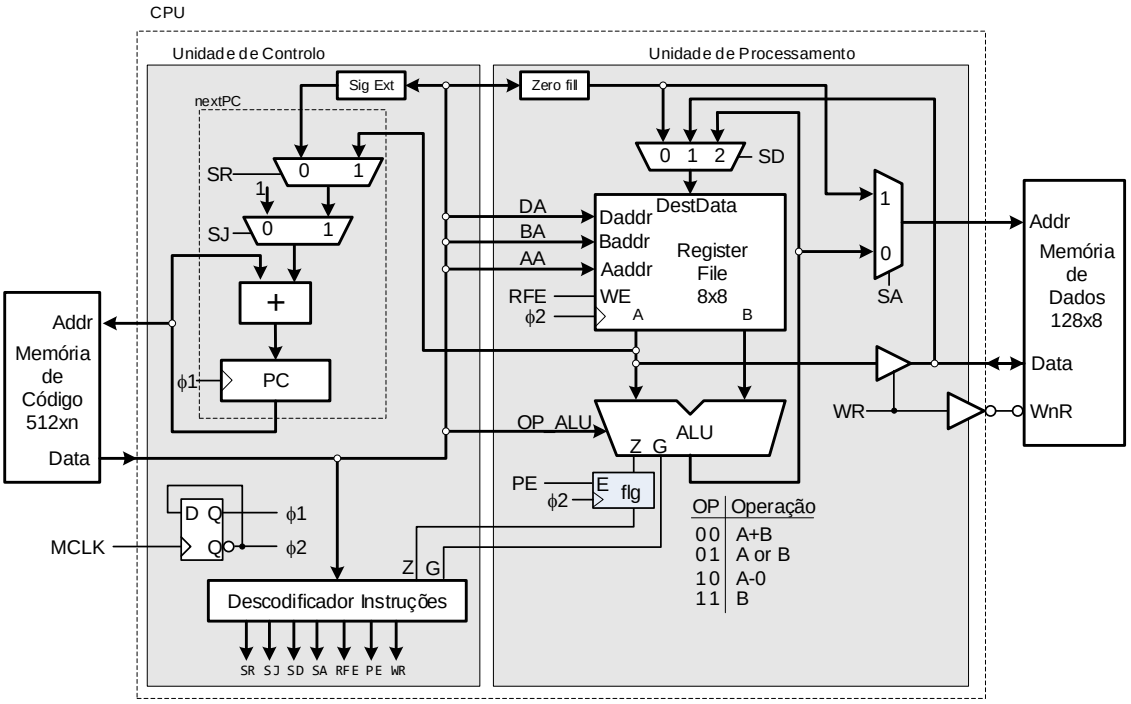
Arquitetura de Computadores

1º Teste (7/jan/2019)

Duração do Teste: 2 horas e 30 minutos

---

[1] Considere um processador, de ciclo único, com o diagrama de blocos apresentado na figura.



O processador suporta a execução do seguinte conjunto de instruções, em que a constante $\texttt{const}_3$ representa um número natural e a constante `offset` representa um número relativo:

N.º	Instrução	Codificação										Descrição
		b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
1	`ldi rx,#const3`	0	1	0	0	rx_2	rx_1	rx_0	c_2	c_1	c_0	`rx = const3`
2	`ld rx,[ry,rx]`	A definir										`rx = M[ry+rx]`
3	`st rx,[ry]`	0	1	1	0	rx_2	rx_1	rx_0	ry_2	ry_1	ry_0	`M[ry] = rx`
4	`add rx,ry`	0	0	0	0	rx_2	rx_1	rx_0	ry_2	ry_1	ry_0	`rx = rx + ry`
5	`orl rx,ry`	0	0	1	0	rx_2	rx_1	rx_0	ry_2	ry_1	ry_0	`rx = rx ∨ ry`
6	`jmp offset`	A definir										`PC = PC + offset`
7	`jz rx`	A definir										`(Z == 1) ? PC = PC + rx : PC = PC + 1`

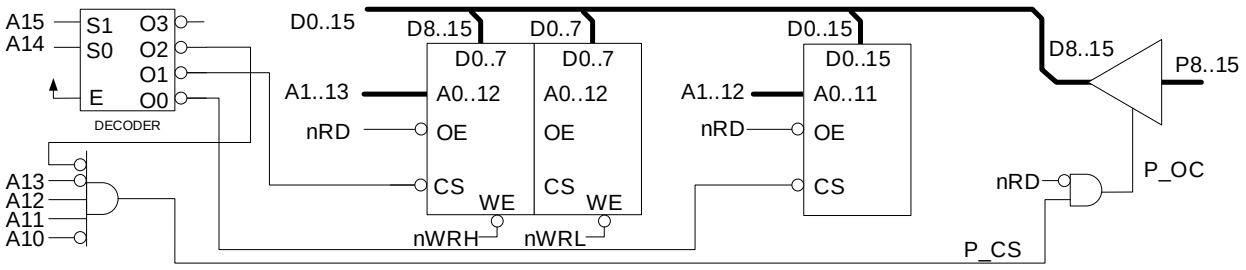
a) Codifique as instruções `ld`, `jmp` e `jz` utilizando uma codificação linear a 4 bits. Explicite os bits do código de instrução que correspondem aos sinais AA, BA, DA, OP_ALU e OPCODE. [2,5 val.]

b) Considerando que o módulo `Descodificador Instruções` é implementado usando exclusivamente uma ROM, indique a programação da mesma. [2,0 val.]

c) Proponha, justificando, um diagrama lógico para o módulo `sigExt`. [0,5 val.]

[Escreva aqui]

[2] Considere o sistema computacional baseado no PDS16 representado na figura.



a) Desenhe o mapa de endereçamento do sistema (incluindo as modificações que vier a realizar nas alíneas seguintes), indicando a funcionalidade, as dimensões, os endereços de início e de fim do espaço atribuído a cada dispositivo, inscrevendo igualmente, se for o caso, a ocorrência de *fold-back*. [2 val.]

b) Adicione uma nova RAM de 8 Kbytes e uma nova ROM de 8 Kbytes utilizando circuitos integrados de 8K*8 e os circuitos necessários para completar a seleção de endereços. [2 val.]

c) Adicione um porto de saída de 16 bits com acesso a 8 e a 16 bits. [1 val.]

[Escreva aqui]

[3] Considere a seguinte função expressa em linguagem C:

```
uint16 binaryStrToInt(uint8[] str, uint8 len) {
  uint16 dec_value = 0;
  uint16 base = 1;

  do {
       len--;
       if (str[len] == '1')
             dec_value += base;
       base = base << 1;
  } while ( len != 0);

  return dec_value;
}
```

a) Traduza para linguagem *assembly* do PDS16 a função `binaryStrToInt` que calcula e retorna o valor numérico representado pelos algarismos binários contidos numa *string*, codificada em ASCII e terminada por 0. Defina as variáveis que entender necessário. [2,5 val.]

b) Considere as definições seguintes e a função main.

```
uint8 num[] = "10101001"
uint16 res;

void main( void ){
      res = binaryStrToInt( num, 8 );
}
```

Traduza para linguagem *assembly* do PDS16 as definições referias e a função main [2,5 val.]

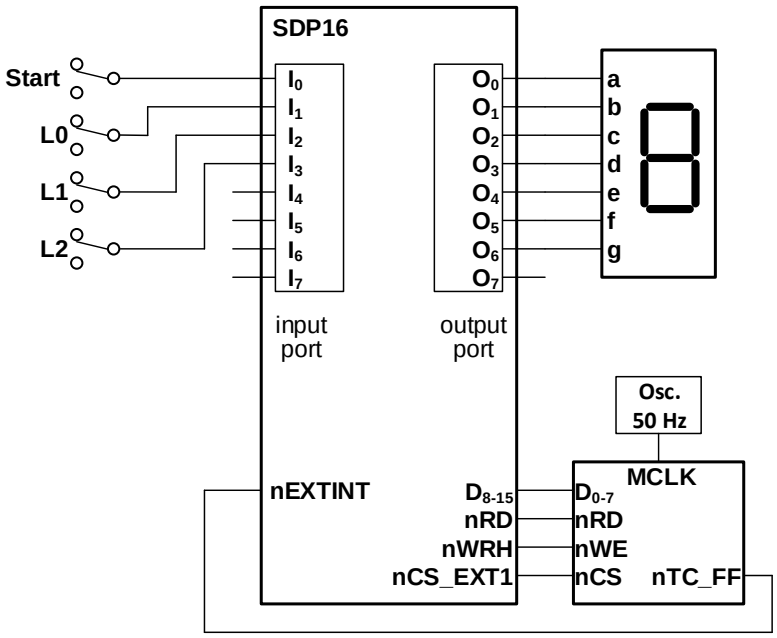
Notas:

1. Com vista ao alojamento de variáveis, assuma que a secção "`.data`" está localizada na zona de memória acessível com endereçamento direto.

2. Na programação em *assembly* deve usar as seguintes convenções: os parâmetros das funções são passados em registos, ocupando a quantidade necessária, pela ordem `r0`, `r1`, `r2` e `r3`; o valor de retorno de uma função, caso exista, é devolvido em `r0`; `int8` e `int16` significam valores inteiros com sinal representados a 8 e a 16 bit, respetivamente; `uint8` e `uint16` significam valores inteiros sem sinal representados a 8 e a 16 bit. A função preserva os registos que utiliza para além dos usados para parâmetros.

[Escreva aqui]

[4] Pretende-se realizar um cronómetro baseado no sistema SDP16 e nos periféricos indicados na figura, com a seguinte definição: a unidade de tempo é o minuto, o contador evolui em sentido crescente, a contagem é iniciada com uma transição de 0 para 1 no botão **Start**; o contador para quando atinge o valor definido em base 2 nas entradas **L0-2**. As seguintes funções servirão como primitivas para a composição do programa de controlo. Programe estas funções em *assembly* do PDS16, segundo as especificações em cada alínea.



a) A função **`uint8_t get_limit()`** devolve o valor representado em binário nas entradas **`L0-2`**. [1 val.]

b) A função **`void get_start()`** aguarda por uma transição de 0 para 1 do botão **`Start`** [1 val.]

c) A função **`void display_write(uint8_t value)`** afixa no *display* o valor passado em parâmetro. Utilize o *array* **`bin7seg`** para a conversão de **`value`** representado em base 2 para a representação em *display* de 7 segmentos. [1 val.]

```
bin7seg: .byte 0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f
```

d) A função **`void timer_init()`** programa o *timer* de modo que o sinal **`nTC_FF`** apresente uma frequência de 2Hz. [1 val.]

e) A rotina de atendimento de interrupção que se encarrega de fazer evoluir a contagem e atualizar o valor mostrado no *display*. [1 val.]