INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

LEIC, LEETC

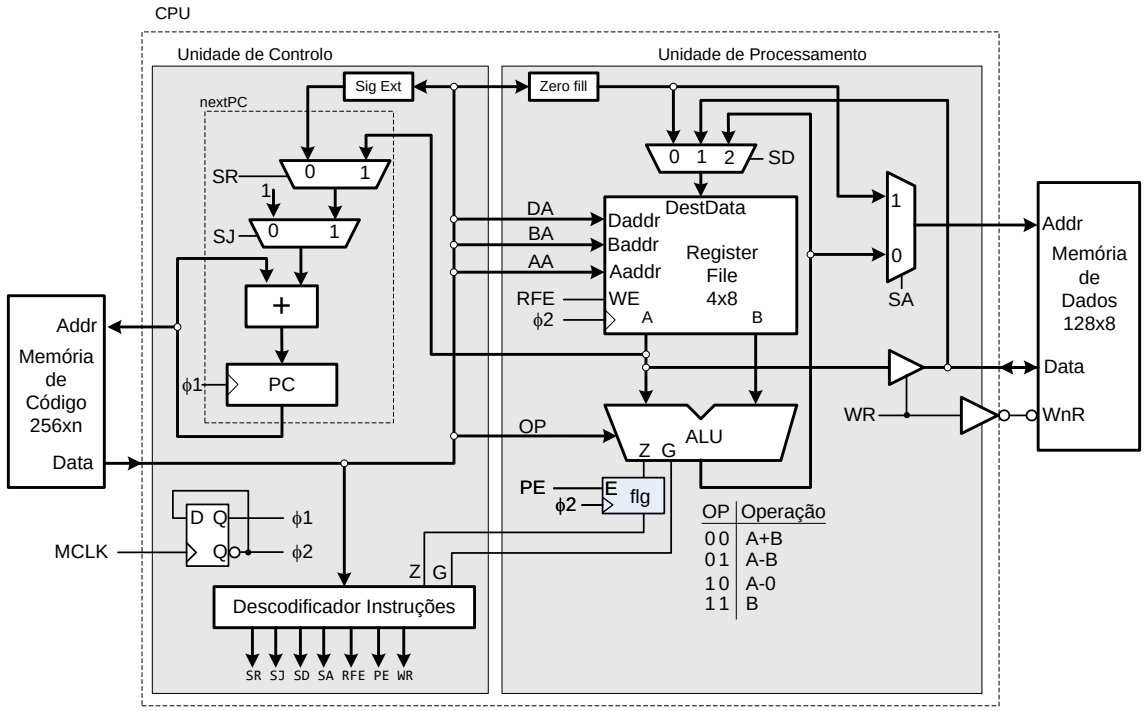
Arquitetura de Computadores

Teste de Época Especial (12/fev/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 CPU suporta a execução do seguinte conjunto de instruções, codificadas com um tamanho fixo:

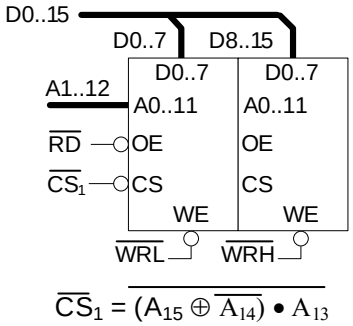
N.º	Instrução	Codificação									Descrição
		b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
1	`ld rx, direct3`	0	0	0	d_2	d_1	d_0	rx_2	rx_1	rx_0	`rx = M[direct3]`
2	`st ry, [rx]`	A definir									`M[ry] = rx`
3	`add rx, rx, ry`	1	0	0	ry_2	ry_1	ry_0	rx_2	rx_1	rx_0	`rx = rx + ry`
4	`jnz offset`	A definir									`(Z == 0) ? PC = PC + offset : PC = PC + 1`
5	`jz rx`	A definir									`(Z == 1) ? PC = PC + rx : PC = PC + 1`

a) Codifique as instruções 3, 4 e 5, procurando maximizar a dimensão da constante `offset`. Explicite os bits do código de instrução que correspondem aos sinais AA, AB, AD, OP e OPCODE. [2,0 val.]

b) Considerando que o módulo **Descodificador de Instruções** é implementado usando exclusivamente uma ROM, indique a programação da mesma para todas as instruções. [2,0 val.]

c) Indique o tamanho da memória de código e o da ROM do **Descodificador de Instruções**. [1,0 val.]

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



Considere um sistema baseado no processador PDS16 com uma memória cuja descodificação é ilustrada na figura. Pretende-se adicionar ao sistema 4K*16 de memória ROM e 16 Kbytes de RAM com acesso ao *byte* e à *word* e um porto de saída de 16 bits com acesso ao *byte* e à *word*.

a) Desenhe o mapa de memória do sistema de acordo com a memória ilustrada e com os dispositivos a adicionar. Indique os endereços de início e de fim de cada zona e eventual existência de *fold-back*. [2.0 val.]

b) Com componentes RAM e ROM de 8K*8, desenhe a nova memória de acordo com o mapa proposto. [2.0 val.]

c) Desenho o porto de saída, de acordo com o mapa da alínea a). [1 val.]

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

```
int8 countDiff (uint8[] str1, uint8[] str2)
{
  uint8 count = 0;
  uint8 idx1 = 0;

  while (str1[idx1] != 0) {
       if (str1[idx1] != str2[idx1])
            count++;
       idx1++;
  }
  return count;
}
```

a) Traduza para linguagem *assembly* do PDS16 a função `countDiff()` que compara a *string* str1 com a *string* str2. A função countDiff () retorna o número de caracteres diferentes entre as duas strings. As *string* estão codificadas em ASCII e terminam com 0. Defina as variáveis que entender necessário [2,5 val.]

b) Considere as definições seguintes e a função main.

```
uint8 stx1[] = "Arquitetura";
uint8 stx2[] = "Arquitectura";
uint8 diff;

void main( void ){
      diff = countDiff( stx1, stx2 );
}
```

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

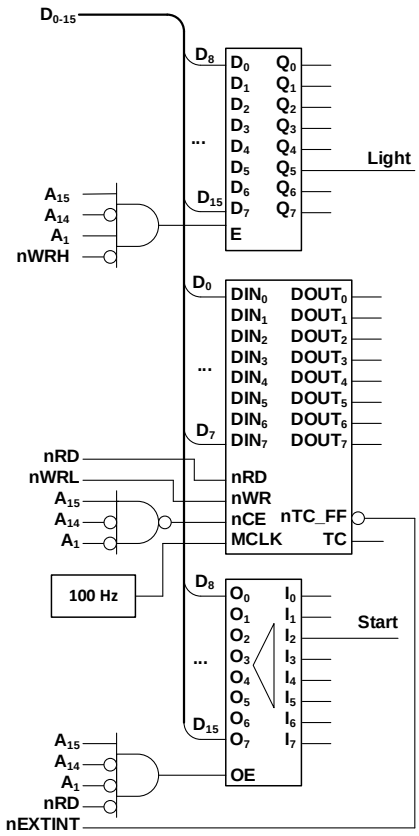
Notas:

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] Considere o seguinte conjunto de portos ligados ao PDS16. Após detetar uma transição no sinal de Start, o circuito deve ativar a saída Light durante 5 segundos, findo o qual deve desligar e esperar pela próxima transição de Start.



Implemente em linguagem *assembly* do PDS16 o componente de programa definido em cada uma das alíneas.

a) A função **`uint8_t get_start()`** que devolve 1 se for detetada transição de 0 para 1 no sinal **`Start`** e devolve 0 no caso contrário. A função não deve ficar bloqueada à espera da transição. [1 val.]

b) A função **`void output_signal()`** que afeta a saída **`Light`** de acordo com a especificação do circuito, sem afetar os restantes bits do porto de saída. Enquanto estiver ligado, deve ignorar a entrada **`Start`** [1 val.]

c) A função **`void prog_timer(unit8 value)`** que configura o temporizador com o valor **`value`**. [1 val.]

d) A rotina de atendimento de interrupção que se encarrega de contar os segundos. [1 val.]

e) O programa principal fazendo uso dos componentes de programa definidos nas alíneas anteriores. [1 val.]