INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

LEETC, LEIC, LEIRT

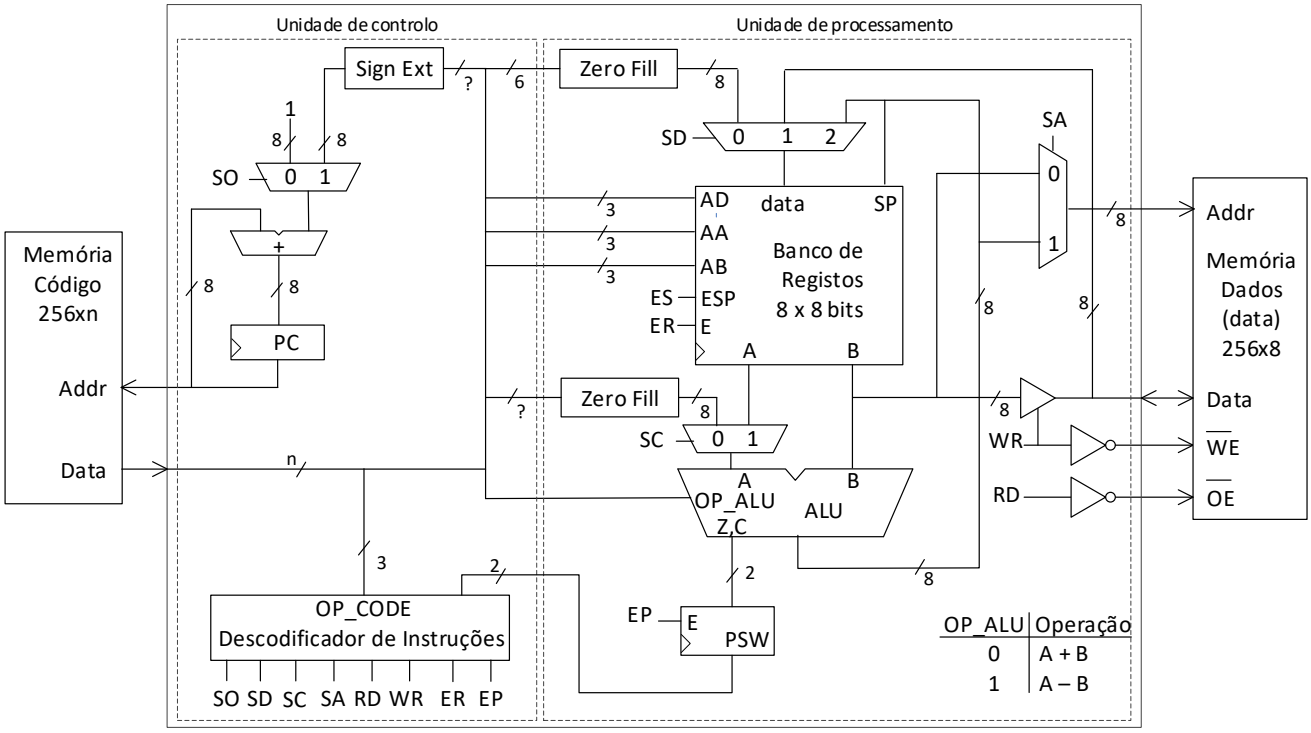
Arquitetura de Computadores

Teste de Época Especial (29/07/2019) *

Duração do Teste: 2 horas e 30 minutos

---

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



Este processador suporta a execução do seguinte conjunto de instruções, em que a constante `imm6` representa um número natural e a constante `offset` representa um número relativo. O *stack pointer*, `SP`, está guardado no registo `R0`.

N.º	Instrução	Codificação												Descrição
		b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
1	`ldr rx, [ry, rz]`	A definir												`rx = M[ry + rz]`
2	`inc rx, ry`	A definir												`rx = ry + 1`
3	`sub rx, ry, rz`	1	0	0	rz_2	rz_1	rz_0	ry_2	ry_1	ry_0	rx_2	rx_1	rx_0	`rx = ry – rz`
4	`pop rx`	A definir												`rx = M[++SP]`
5	`mov rx, #imm6`	1	1	0	i_5	i_4	i_3	i_2	i_1	i_0	rx_2	rx_1	rx_0	`rx = imm6`
6	`blo offset`	A definir												`(C==1) ? PC = PC + offset` `: PC = PC + 1`

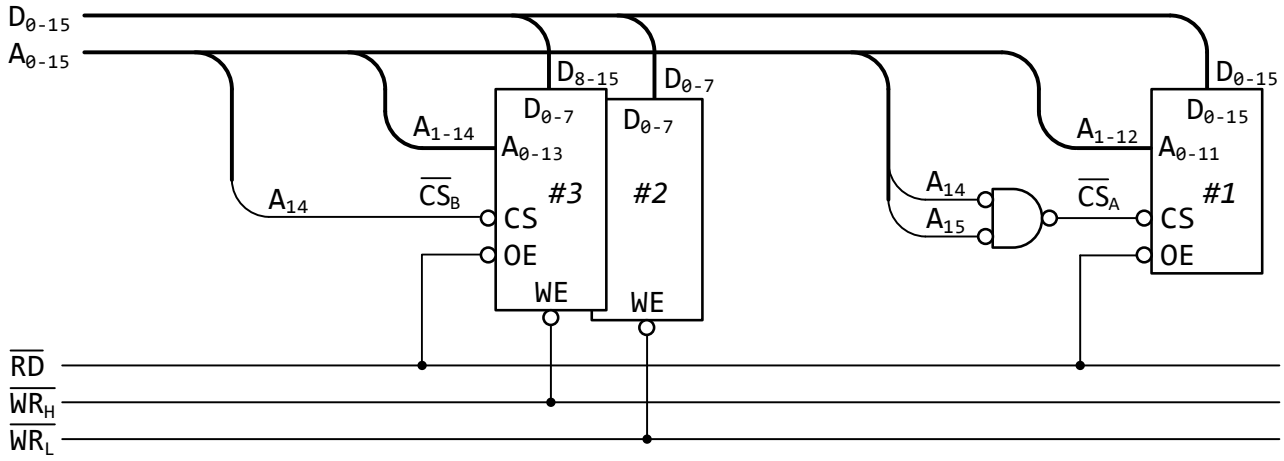
a) [2,0 val] Codifique as instruções **`ldr`**, **`inc`**, **`pop`** e **`blo`**, utilizando codificação linear a três bits e definindo adequadamente a dimensão da constante **`offset`**. Explicite os bits do código de instrução que correspondem aos sinais AA, AB, AD, OP_ALU e OP_CODE

b) [2,0 val] Considerando que o módulo `Descodificador de Instruções` é implementado usando exclusivamente uma ROM, indique a programação da mesma. Indique a dimensão, em bits, dessa memória.

c) [1,0 val] Pretende-se acrescentar ao conjunto de instruções do processador a instrução **`ldr rx, indirect6`** que realiza a operação **`rx = M[PC + indirect6]`**, em que **`indirect6`** representa um número natural. Indique as alterações a realizar no diagrama de blocos do processador para que este passe a suportar a execução da nova instrução.

* Enunciado alterado para corrigir gralha na pergunta 1 do enunciado original.

[2] Considere o diagrama da figura abaixo que descreve um exemplo de descodificação de endereços, correspondente ao mapeamento destes dispositivos de memória, em torno de um processador P16.



a) [1,0 val] Indique os tipos e as dimensões (endereços e dados) dos dispositivos #1 a #3, individualmente tomados, e as capacidades (em quilobytes – KB) dos módulos que formam.

b) [1,0 val] Desenhe o mapa de endereçamento do conjunto, indicando os tipos, as dimensões e os endereços de início e de fim do espaço atribuído a cada módulo, inscrevendo igualmente, se for o caso, a ocorrência de subaproveitamento ou de *fold-back* e a localização de eventuais zonas livres e de zonas interditas (também designadas por "conflito").

c) [1,5 val] Continue o exemplo da tabela abaixo, para as instruções nas linhas 24 a 30 do troço de código dado, completando na sua folha de teste o registo da atividade dos barramentos e dos sinais em referência, observados passo-a-passo (*single step*) durante a sua execução, admitindo que o código é executado sobre o sistema apresentado na figura. A listagem foi produzida pelo *Assembler* PAS v1.2.2.

```
21
22 1000 5065  mov  r0, 0x55
23 1002 1020  str  r0, [r1]
24 1004 2320  str  r3, [r2]
25 1006 24AD  sub  r4, r2, #0xA
26 1008 4300  ldr  r3, [r4]
27 100A 0278  movt r2, 0x80
28 100C 2508  ldrb r5, [r2, 0]
29 100E A608  ldrb r6, [r2, 1]
30 1010 FF5B  b    .
31
```

CTRL			ADDR	DATA	*Instruction*
nWRH	nWRL	nRD	A15 … A0	D15 … D0	
H	H	L	1000	6550	`mov r0, 0x55`
H	H	L	1002	2010	`str r0, [r1]`
L	L	H	FFFE	0055	
			1004		

Exemplo para copiar e completar.
Atividade dos barramentos observados passo-a-passo,
com os seguintes valores iniciais:
`r0 = 0x0000; r1 = 0xFFFE; r2 = 0x0008;`
`r3 = 0xABCD; r4 = 0x7342; pc = 0x1000.`

Nota – genericamente, no barramento de dados pode ocorrer: um valor concreto; alta impedância – `ZH`; ou conflito – `conf`.

d) [1,5 val] Pretende-se reformular a seleção de endereços, cumprindo os seguintes critérios:

- Não há zonas interditas (de "conflito");
- A dimensão do espaço atribuído a cada memória é coincidente com a sua capacidade;
- As memórias do mesmo tipo ficam em endereços contíguos entre si;
- A zona de ROM tem início no endereço `0x0000`;
- Reserva-se uma área de 8 KB para futura expansão da zona de ROM;
- O espaço livre restante deve permitir a colocação de um porto de saída a 16 bits, apenas com acesso *word-wise*, com dimensão igual ou inferior a esse espaço.

Desenhe o novo mapa de endereçamento e apresente as expressões lógicas dos novos sinais de *chip select*.

[3] Atente as seguintes implementações das funções `min`, `swap` e `selection_sort`, em que os tipos `int16_t` e `uint16_t` representam valores inteiros e naturais, respetivamente, codificados com 16 bits e o tipo `int8_t` representa valores inteiros codificados com 8 bits.

```
int16_t min( int8_t v[], uint16_t idx1, uint16_t idx2 ) {

   if( v[idx1] < v[idx2] )
        return idx1;
   else
        return idx2;
}

void swap( int8_t v[], uint16_t idx1, uint16_t idx2 ) {
  int8_t temp;

   temp = v[idx1];
   v[idx1] = v[idx2];
   v[idx2] = temp;
}

void selection_sort( int8_t v[], uint16_t n ) {
   uint16_t i, j, m;

   for ( i = 0; i < n-1; i++ ) {
        m = i;
        for ( j = i+1; j < n; j++ ) {
              m = min( v, j, m );
        }
        swap( v, m, j );
   }
}
```

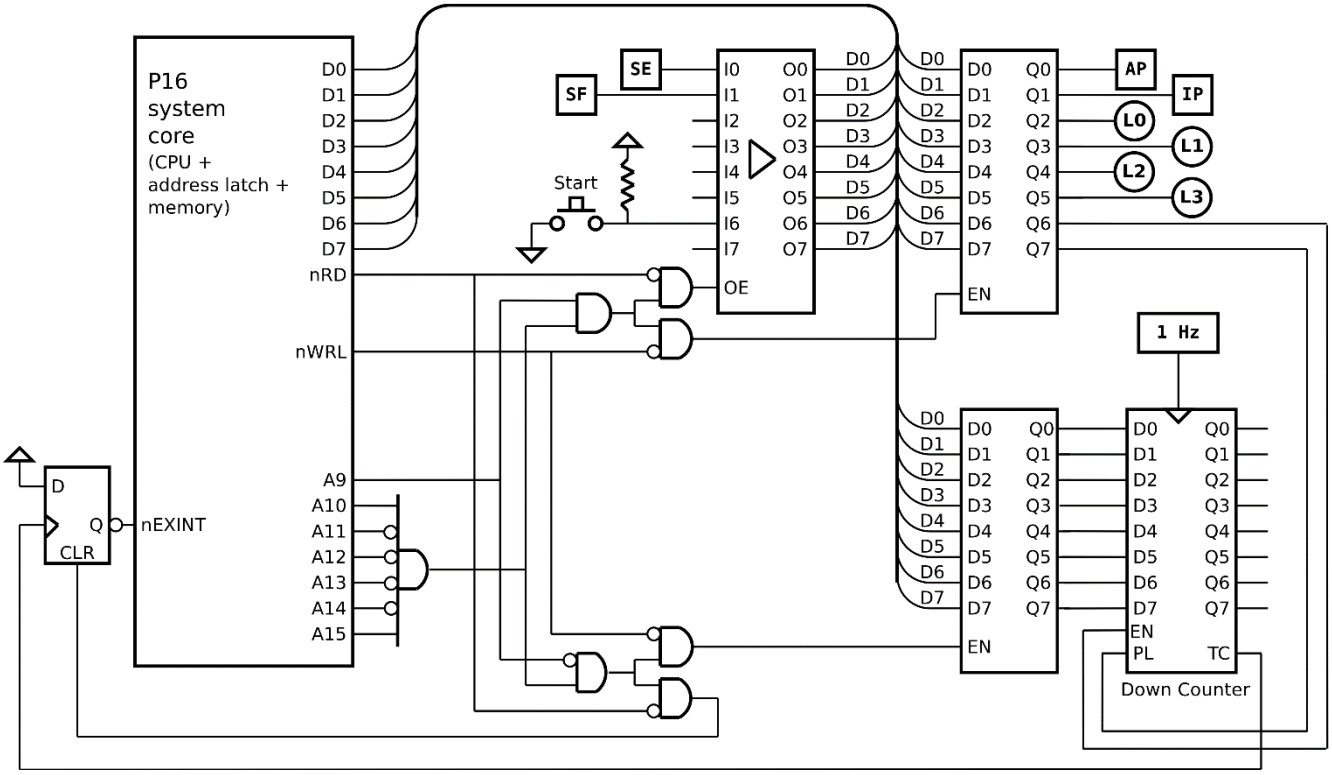
Considerando que *i*) os parâmetros das funções são passados nos primeiros quatro registos de uso geral do processador, o primeiro em `r0`, o segundo em `r1` e assim sucessivamente, *ii*) o valor de retorno das funções é devolvido no registo `r0`, *iii*) as funções devem preservar o conteúdo dos registos que utilizam para além dos convencionados para passagem de parâmetros (`r0` a `r3`) e *iv*) o *stack pointer* foi devidamente inicializado,

a) [1,5 val] Implemente a função `min` na linguagem *assembly* do P16, definindo, se necessário, as variáveis auxiliares utilizadas.

b) [1,0 val] Implemente a função `swap` na linguagem *assembly* do P16, definindo, se necessário, as respetivas variáveis.

c) [2,5 val] Implemente a função `selection_sort` na linguagem *assembly* do P16, definindo, se necessário, as respetivas variáveis.

[4] Para automatizar um sistema de rega, pretende-se implementar um sistema de controlo recorrendo ao processador P16 e ao *hardware* indicado na figura.



Neste sistema, os sensores `SE` e `SF` são utilizados para aferir a quantidade de água disponível no tanque de rega, os quatro LED (`L0` a `L3`) informam sobre essa quantidade e os atuadores AP e IP são utilizados para controlar o funcionamento das bombas de admissão e de rega, respetivamente. O botão `Start` serve para iniciar um novo período de rega, que tem uma duração aproximada de três minutos.

Com vista à realização global deste sistema:

a) [1,5 val] Escreva a rotina que faz a iniciação do contador decrescente (`Down Counter`) para uma nova contagem, de modo a que a sua saída `TC` fique ativa no final do período definido para uma rega.

b) [1,5 val] Escreva a rotina que avalia o estado do reservatório por inspeção dos sensores `SE` e `SF`, retornando 0 quando o reservatório está vazio, 2 quando está cheio e 1 quando o nível de água se encontra entre estes dois limites.

c) [1,5 val] Escreva a rotina que põe em funcionamento a bomba de admissão (`AP`), sem comprometer o funcionamento da outra bomba (`IP`) ou o valor afixado nos LED.

d) [1,5 val] Escreva a rotina para o atendimento da interrupção (`ISR`), responsável por terminar o período de rega em curso.

Nota: Defina todos os símbolos e variáveis que entender necessários para a realização das alíneas a) a d).