INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

LEETC, LEIC, LEIRT

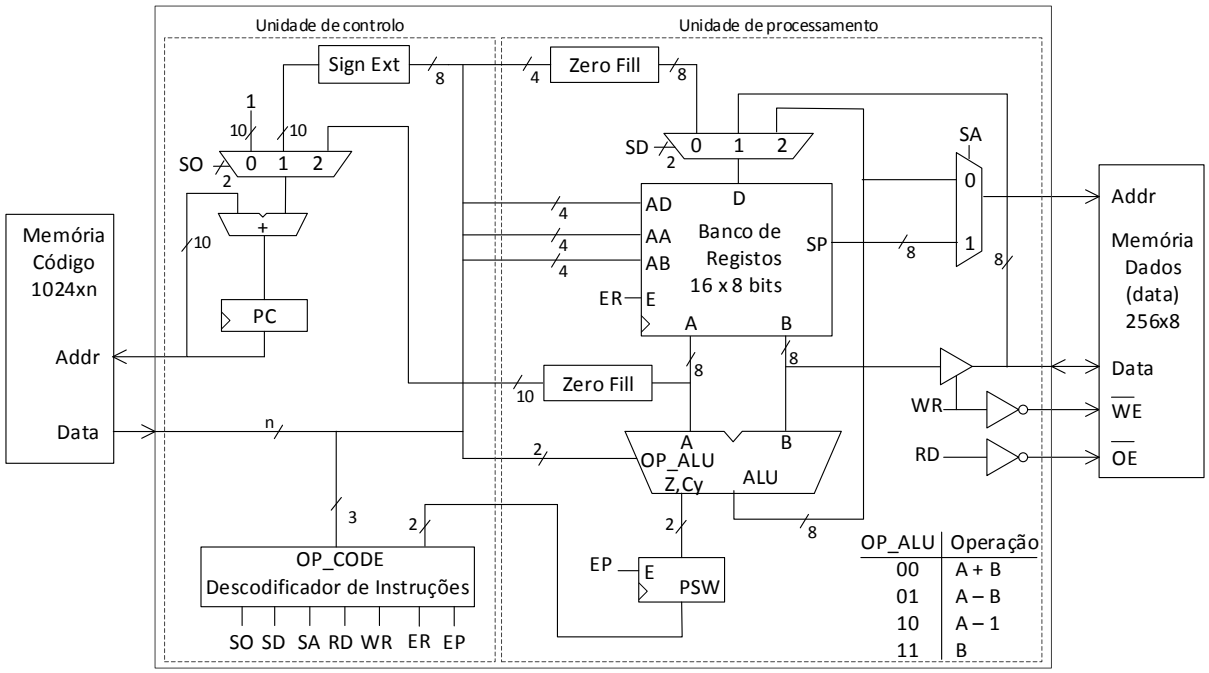
Arquitetura de Computadores

1.º Teste (27/06/2019)

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 $\texttt{imm}_4$ representa um número natural e a constante `offset8` representa um número relativo. O *stack pointer*, `SP`, está guardado no registo `R13`.

N.º	Instrução	Codificação											Descrição
		b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
1	`mov rx,#imm4`	0	0	0	rx_3	rx_2	rx_1	rx_0	i_3	i_2	i_1	i_0	`rx = imm4`
2	`ldr rx,[ry]`	A definir											`rx = M[ry]`
3	`cmp rx,ry`	A definir											`rx – ry`
4	`add rx,ry`	1	0	0	rx_3	rx_2	rx_1	rx_0	ry_3	ry_2	ry_1	ry_0	`rx = rx + ry`
5	`push rx`	A definir											`M[SP--] = rx`
6	`bcc offset8`	0	0	1	o_7	o_6	o_5	o_4	o_3	o_2	o_1	o_0	`(C==0) ? PC = PC + offset8 : PC = PC + 1`
7	`beq rx`	A definir											`(Z==1) ? PC = PC + rx : PC = PC + 1`

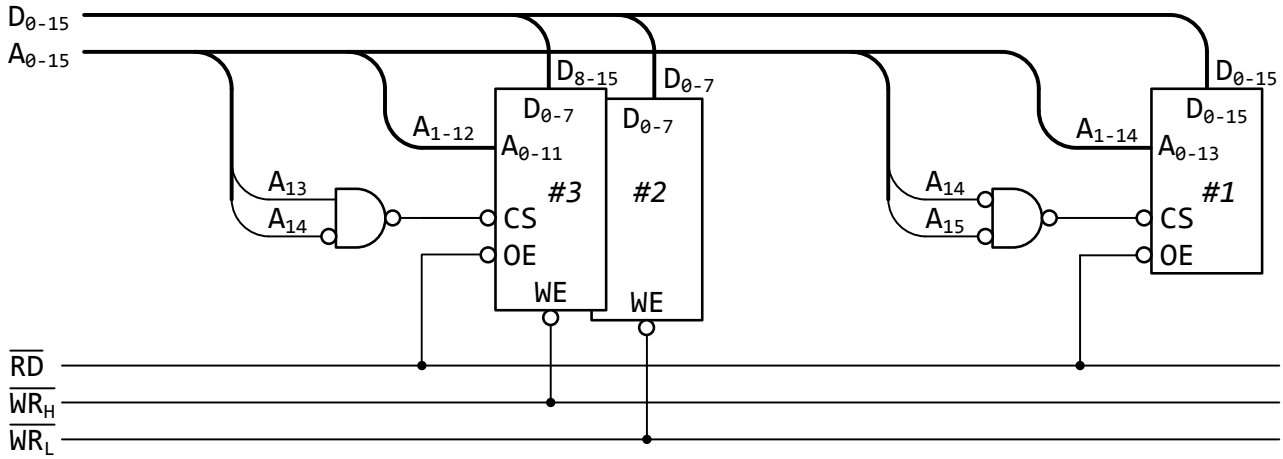
a) [2,0 val] Codifique as instruções `ldr`, `cmp`, `push` e `beq` utilizando codificação linear a 3 bits. Explicite os bits do código de instrução que correspondem aos sinais AA, AB, AD, OP_ALU e OP_CODE.

b) [1,5 val] Considerando que o módulo `Descodificador de Instruções` é implementado usando exclusivamente uma ROM, indique a programação da mesma.

c) [0,5 val] Considerando `PC = 0x0A0`, indique a gama de endereços alcançáveis com a instrução `bcc offset8`.

d) [1,0 val] Sabendo que a instrução `cmp rx, ry` será substituída pela instrução `cmp rx, imm4`, que realiza a operação `rx - imm4`, indique as alterações a realizar no diagrama de blocos do processador para que este passe a suportar a execução da nova instrução.

[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) [0,5 val] Identifique os tipos e indique as dimensões dos dispositivos #1 a #3, individualmente tomados.

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

c) [1,5 val] Continue o exemplo da tabela abaixo, 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 execução do troço de código dado, admitindo que o código é executado sobre o sistema. A listagem foi produzida pelo *Assembler* PAS v1.2.2.

```
31          do_this:
32 1234 1011     ldr  r0, [r1, r2]
33 1236 03B0     mov  r3, r0
34 1238 A00C     ldr  r0, var1_addr
35 123A 1231     str  r2, [r1, r2]
36 123C 0258     b    and_that
37 123E 0167     mov  r1, 0x0070
38 1240 0175     movt r1, 0x50
39          and_that:
40 1242 8260     mov  r2, 0x0008
41 1244 6270     movt r2, 0x06
42 1246 4228     strb r2, [r4]
43 1248 4220     str  r2, [r4]
44 124A 4000     ldr  r0, [r4]
45 124C FF5B     b    .
46          var1_addr:
47 124E 2030     .word    0x3020
48
```

CTRL			ADDR	DATA	*instruction*
nWRH	nWRL	nRD	A15 … A0	D15 … D0	
H	H	L	1234	1110	`ldr r0, [r1, r2]`
H	H	L	1244	7062	
H	H	L	1236	B003	`mov r3, r0`
			1238		

Exemplo para copiar e completar.
Atividade dos barramentos observados passo-a-passo,
com os seguintes valores iniciais:
`r0 = 0x5070; r1 = 0x1040; r2 = 0x0204;`
`r3 = 0xFEDC; r4 = 0x2030; pc = 0x1234.`

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

d) [1,5 val] Redesenhe o sistema proposto, apresentando o novo mapa de endereçamento, mantendo o mapeamento da memória ROM e acrescentando: 32 KB de memória do tipo RAM, um porto de entrada e um porto de saída, ambos a 16 bits. É recomendável que os módulos de memória RAM fiquem contíguos entre si.

e) [0,5 val] Desenhe o diagrama correspondente à instalação da nova memória, conforme a solução realizada na alínea anterior, escolhendo os dispositivos RAM que considerar mais adequados de entre os seguintes: 8 K * 8 | 8 K * 16 | 16 K * 8 | 16 K * 16 | 32 K * 8 | 32 K * 16.

[3] Atente as seguintes implementações das funções `abs_diff` e `sum_abs_diff`, em que o tipo `int16_t` representa valores inteiros codificados com 16 bits e os tipos `uint16_t` e `uint8_t` representam valores naturais codificados, respetivamente, com 16 bits e 8 bits.

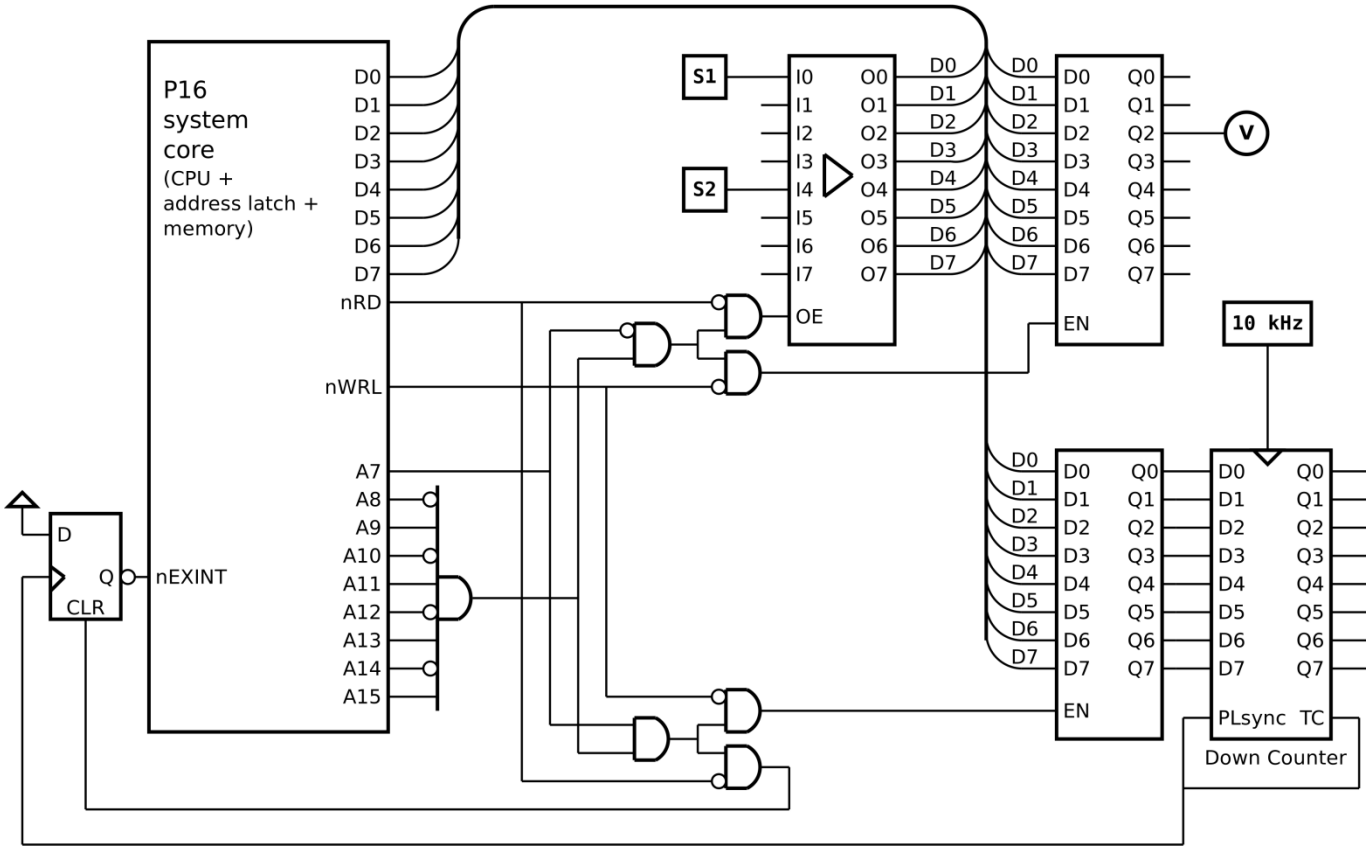
```
uint8_t abs_diff( uint8_t curr_pel, uint8_t ref_pel )
{
    int16_t diff;

    diff = curr_pel - ref_pel;
    if ( diff < 0 )
        diff = -diff;
    return (uint8_t) diff;
}

uint16_t sum_abs_diff( uint8_t curr_blk[], uint8_t ref_blk[], uint8_t blk_width,
                       uint8_t blk_height )
{
    uint16_t acc_diff = 0;
    uint16_t pos = 0;
    uint8_t i, j;

    for ( i=0; i < blk_height; i++ )
        for ( j=0; j < blk_width; j++ )
        {
            acc_diff += abs_diff( curr_blk[pos], ref_blk[pos] );
            pos++;
        }
    return acc_diff;
}
```

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

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

[4] Para promover a acalmia de tráfego na entrada de uma localidade, pretende-se implementar um sistema de controlo de velocidade com semáforo, recorrendo ao processador P16 e ao *hardware* indicado na figura.



Neste sistema, pretende-se que a luz vermelha do semáforo (`V`) seja ativada sempre que um veículo se desloque a uma velocidade superior a 50 km/h na zona de controlo de velocidade. Esta zona é demarcada por dois sensores de presença de veículo, `S1` e `S2`, colocados no solo a 5 m de distância entre si. A luz vermelha do semáforo deverá ficar acesa por 10 s, após o que deverá voltar a apagar.

a) [1,0 val] Escreva uma rotina que coloque o valor lógico presente na entrada `I0` do porto de entrada associado ao sensor `S1` na saída `Q2` do porto de saída associado à luz vermelha do semáforo (`V`).

b) [1,0 val] Escreva uma rotina que faça a programação do contador decrescente do subsistema divisor de frequência (`Down Counter`) de modo a que a sua saída TC apresente uma frequência de 1 kHz.

c) [1,0 val] Escreva a rotina de atendimento da interrupção (ISR) responsável por manter atualizado o valor da variável global `system_clock`, que representa o tempo decorrido em milissegundos.

d) [2,0 val] Escreva um programa que acenda a luz vermelha quando é detetado um veículo a circular com velocidade superior a 50 km/h.

Nota 1: Sabendo-se que a fórmula v = 18000 / t, com t em milissegundos, permite calcular a velocidade do veículo em km/h, resulta que a velocidades superiores a 50 km/h correspondem intervalos de tempo entre atuações dos sensores inferiores a 360 ms.

Nota 2: Defina todos os símbolos e variáveis que entender necessários para a realização das alíneas a) a d).