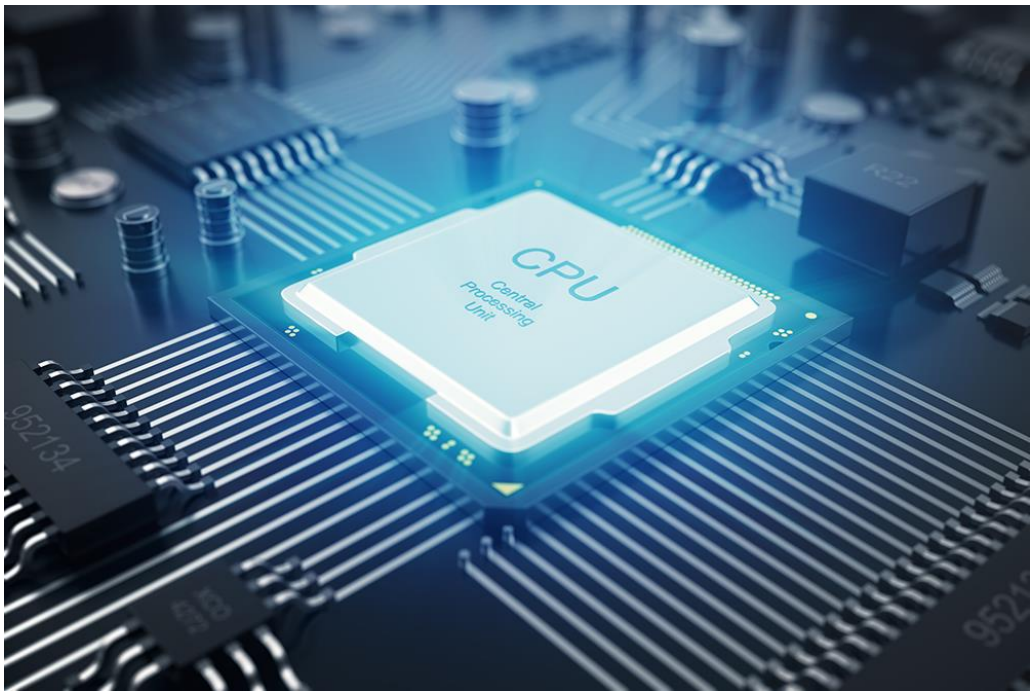
Disciplina de Arquitetura de Computadores
Grupo 07 (4º Trabalho)

- Mihail Arcus (nº50870)
- Paulo Torres (nº50867)

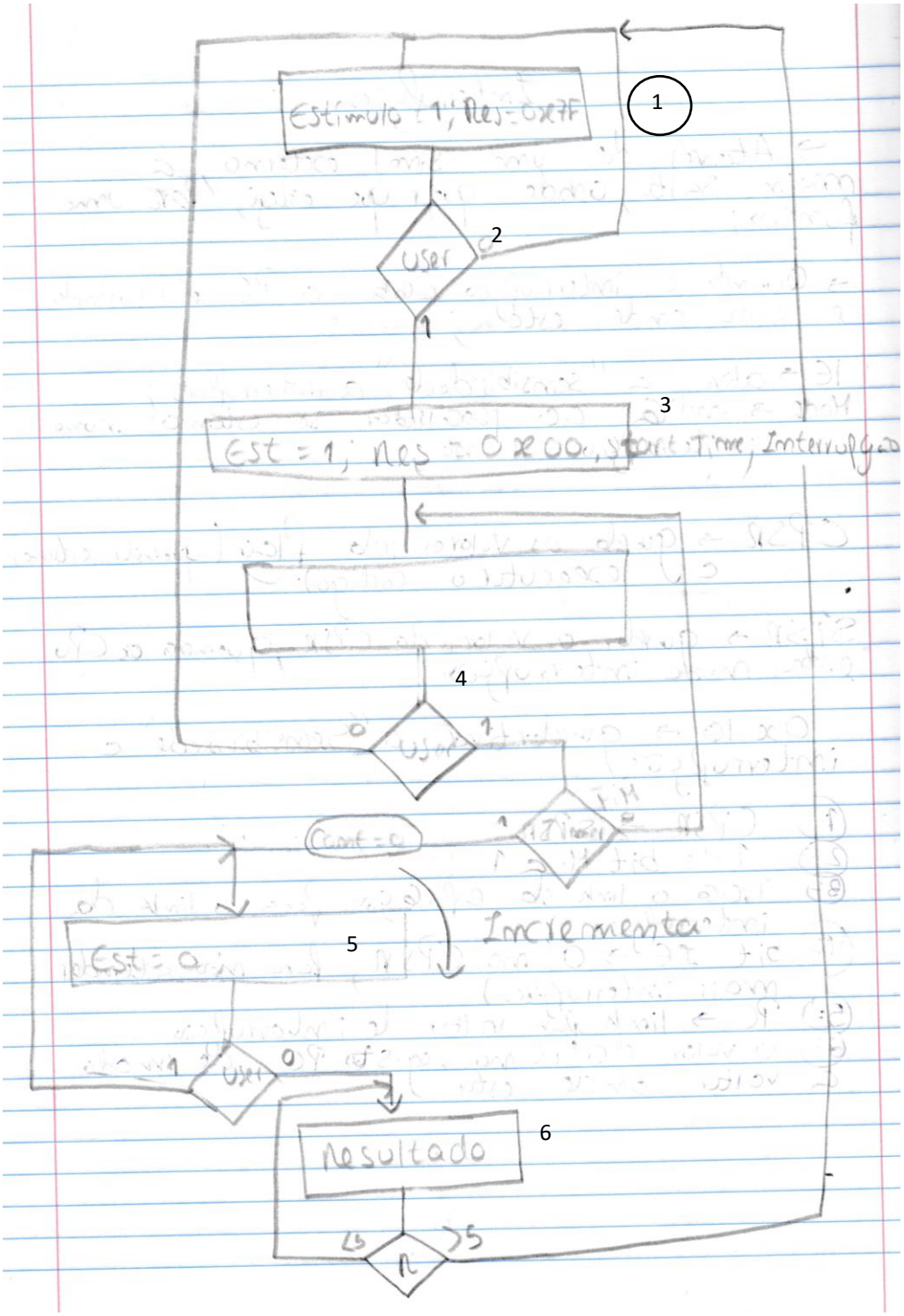


❖Objetivo do trabalho:

Este trabalho consiste em desenvolver um programa que mede o tempo de reação de pessoas a um estímulo visual.

O sistema tem de contar o tempo que demora entre a ativação do estímulo visual e a resposta do utilizador

❖Fluxograma do programa pretendido:



1-> No ínicio do programa, os LED'S do resultado mais o led do estímulo devem estar acesos, para indicar que o programa está pronto a funcionar;

2-> O user (que é o bit I0 do porto de entrada) está continuamente a ser testado, de forma a que se for 0, continua no loop descrito no passo nº 1 (volta ao primeiro estado), e caso seja 1, passamos então para o estado seguinte;

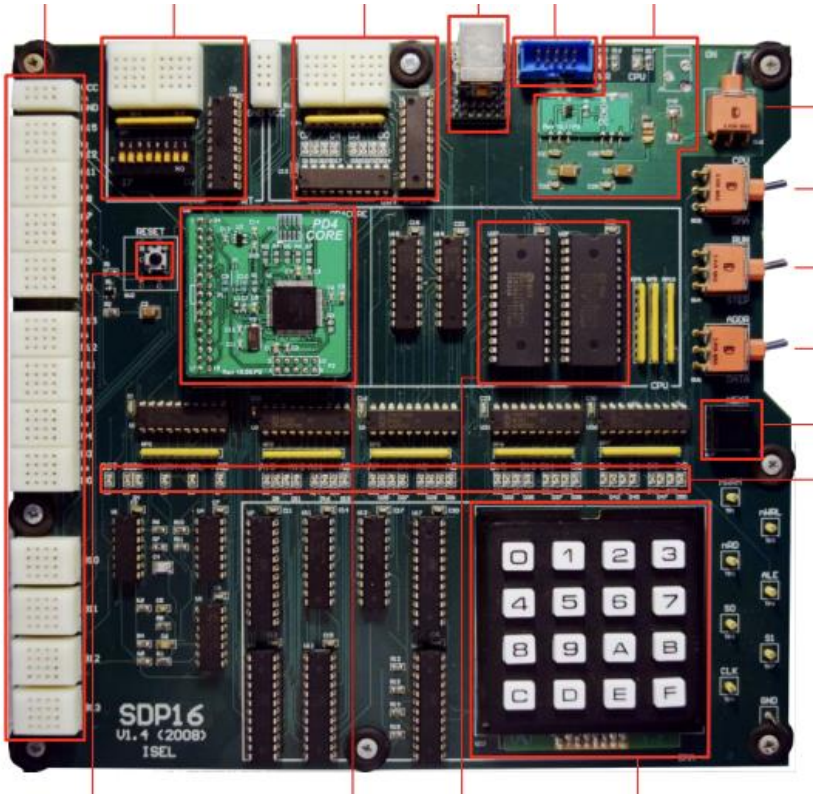
3-> No estado seguinte, a máquina de estados entra numa fase em que conta o tempo que o utilizador colocou no porto de entrada. Para tal, os led's do resultado apagam-se ficando apenas o led STIMULUS ativo; O contador é reiniciado e definido para contar em segundos, e são ativadas as interrupções;

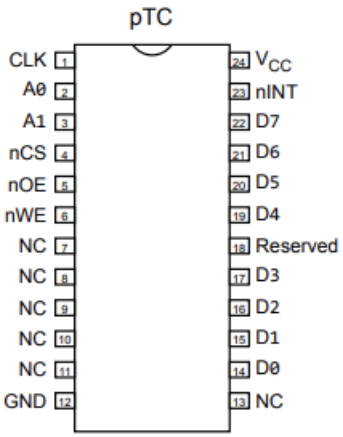
4-> Após entrarmos no estado anterior, o user vai continuar a ser testado. Caso permaneça a 1, verificamos se o sysclk está igual ao timer. Se sim, passamos para o estado seguinte, se não, continuamos neste ciclo até o sysclk ser igual ao timer. Caso o user esteja a 0, o programa é interrompido e volta ao primeiro estado;

5-> Neste estado vai ser então contado o tempo de reação da pessoa ao estímulo visual. Para isso, o contador é reiniciado e definido para contar o tempo em milissegundos. O STIMULUS vai ser colocado a 0. Enquanto o user estiver a 0, a máquina de estados continua neste ciclo (e vai incrementando o contador). Quando o user é colocado a 0 passamos então para o estado seguinte;

6-> Neste estado é afixado o resultado no porto de saída em milissegundos. É também definido para afixar o resultado durante 5 segundos. Passados esses 5 segundos o programa acaba e volta ao primeiro estado.

❖Esquema de ligações:





Para o sistema conseguir contar o tempo, recorremos a um dispositivo externo chamado pTC (Pico Timer/Counter). As ligações que realizamos entre a placa SPD16, o pTC e ainda a placa ATB (que nos vai dar o clock), foram as seguintes:

SDP16	pTC	ATB
-	**CLK** (Pino 1)	**OSCILLATOR** (Linha 3)
A0-A1: T.P.B B5 (Linhas 4 e 3, respetivamente)	**A0-A1** (Pinos 2 e 3)	-
nCS_EXT0: T.P.B B13 Linha 4	**nCS** (Pino 4)	-
nRD: T.P.B B10 Linha1	**nOE** (Pino 5)	-
nWRL: T.P.B B10 Linha 2	**nWE** (Pino 6)	-
GND: T.P.B B1 Linha 2	**GND** (Pino 12)	**GND:** T.P.B P7
D0-D3: T.P.B B9 (Linhas 4 a 1, respetivamente)	**D0-D3** (Pinos 14 a 17)	-
D4-D7: T.P.B B8 (Linhas 4 a 1, respetivamente)	**D4-D7** (Pinos 19 a 22)	-
nINT_EXT: T.P.B B12 Linha 4	**nINT** (Pino 23)	-
VCC: T.P.B B1 Linha 1	**Vcc** (Pino 24)	**Vcc:** T.P.B P4

Legenda: T.P.B -> *Tie-Point Block*

❖Funcionamento do programa:

```
main:
    push r5
main_loop:
;bits todos a 1
    mov r0, #0xFF
    bl outport_write
;User 1 ou 0
    bl inport_read
    mov r1, #1
    and r2, r1, r0
    cmp r2, r1
    bne main_loop
;Inicialização
    mov r3, #0
    ldr r1, SYSCLK_ADDR2
    str r3, [r1,#0]
    mov r0, #1
    bl outport_write
    mov r0, #SYSCLK_FREQ
    bl  ptc_init
    mrs r0, cpsr
    mov r1, #CPSR_BIT_I
    orr r0, r0, r1
    msr cpsr, r0
    mov r3, #0
    ldr r1, SYSCLK_ADDR
    str r3, [r1,#0]
main_loop2:

;Ler user 1 ou 0
    bl inport_read
    mov r1, #1
    and r2, r1, r0
    cmp r2, r1
    bne main_loop

;Ler timer
    bl inport_read
    lsr r0, r0, #4
    mov r5, r0

;Ler variavel global/contador
    ldr r1, SYSCLK_ADDR
    ldrb r3, [r1,#0]
```

Nesta parte inicial do código, estamos no 1º estado, da máquina de estados do fluxograma acima.

Colocamos no porto de saída, usando a função "outport_write", o valor #0xFF, que acende todos os LED'S como pretendido (resultado + STIMULUS todos a 1).

Para testar se o user está a 0 ou a 1 decidimos realizar um and entre o que vem do porto de entrada (utilizámos a função "inport_read") e 1. Colocámos o 1 num registo temporário, fizemos o and entre o r0 e o r1 e guardamos no r2. Em seguida fizemos uma comparação entre o registo 2 e o registo 1, e caso fossem diferentes (ou seja, o user for 0), fazemos um salto para o main_loop (onde recomeça este loop).

Aqui entramos no 2º estado da máquina de estados.

Reiniciamos o sysclk colocando-lhe o valor 0. Para tal carregamos para o r1 o endereço do sysclk, e guardamos na primeira posição do r1, o valor que definimos no r3 (#0).

Deixamos o STIMULUS ligado, mas desligamos os led's do resultado, colocando no porto de saída o valor 1 (pois o STIMULUS é o primeiro bit do porto de saída).

Carregamos para o ptc o valor de frequência definido na diretiva .equ (0x000A), o que vai fazer com que o pTC conte até 10, antes de entrar na interrupção, que incrementa o sysclk. Ao contar até 10, e como definimos o clock a 10Hz, ele vai demorar 100ms a contar cada bit, perfazendo 1 segundo ao contar os 10 bits(pois é o que pretendemos nesta fase do programa. Ele tem de contar de segundo em segundo até chegar ao timer definido pelo utilizador). As interrupções são ativadas.

Nesta fase do programa estamos a testar se o user continua ativo. Para tal repetimos o procedimento descrito acima: fazemos um and entre o que vem do porto de entrada (r0) e 1. Guardamos o resultado do and no r2. Em seguida fazemos uma comparação entre o registo 2 e o registo 1, e caso fossem diferentes (ou seja, o user for 0), fazemos um salto para o main_loop (ou seja, o programa é interrompido).

Aqui lemos os bits que estão no "inport_read"e realizámos um deslocamento de 4 bits para a direita e guardámos o resultado no registo r5, pois assim facilita a comparação do tempo que o utilizador definiu com a variável global que conta o tempo.

Aqui nós lemos o que está na variavel global e guardamos no registo r3.

```
;Compara variavel global com timer
    cmp r3, r5
    Blo main_loop2

;Inicializar o contador
    push r4
    mov r3, #0
    mov r4, #1

;Contador a 0
    ldr r1, SYSCLK_ADDR
    strb r3, [r1,#0]

    bl ptc_stop
    mov r0, #SYSCLK_FREQ2
    bl ptc_init

;Estimulo a 0
    mov r0, #0
    bl outport_write

loop:
;Fica em loop enquanto user for 1
    bl inport_read
    and r5 , r4, r0
    cmp r5, r4
    beq loop
;Vai buscar o valor do contador e apresenta o tempo do estimulo a 0,1 segundos
    ldr r1, SYSCLK_ADDR
    ldrb r0, [r1, #0]
    lsl r0, r0, #1
    bl outport_write
    pop r4
    pop r5
;Reniciar o temporizador a cada segundo
    bl ptc_stop
    mov r0, #SYSCLK_FREQ
    bl ptc_init
    mov r3, #0
    ldr r1, SYSCLK_ADDR
    strb r3, [r1,#0]
resultado_loop:
;Apresenta o resultado por 5 segundos
    mov r0, #5
    ldrb r2, [r1,#0]
    cmp r2, r0
    blo resultado_loop
    b main
```

Aqui realizámos a tal comparação dos registos. Caso eles sejam iguais significa que o tempo do temporizador chegou ao tempo do utilizador, assim continuando o código, caso contrário, volta ao loop.

Depois de o temporizador chegar ao tempo do utilizador, reiniciamos a contagem do tempo, pois aqui queremos calcular o tempo de uma maneira mais precisa. Para tal, colocamos o temporizador a entrar na interrupção a cada 0,1s e assim contar a cada 0,1s.

Aqui nós utilizámos uma variável r4 e r3 como auxiliares. Neste caso, utilizámos o r3 para colocar a variável global a 0, para depois voltar a contar o tempo.

Aqui nós paramos o temporizador e voltamos a inicializar com um valor de 1, para assim contar a cada 0,1s.

No final colocamos no outport os bits todos a 0, indicando assim para o utilizador que o tempo para o estimulo já está a contar.

Aqui neste loop, o processador fica em constante loop até que o bit mais à direita do inport read seja 1, indicando assim que o utilizador ativou o estimulo. Para tal, voltamos a fazer o and com 1 e o que vem do inport read.

Aqui, depois de o utilizador dar o estimulo, vamos buscar o valor que está na variável global e realizamos um deslocamento para a esquerda, pois o resultado está nos 7 bits mais à esquerda.

O resultado é então apresentado ao utilizador em forma de 0,1s.

No final do código vamos apresentar o resultado por 5 segundos. Para tal, reiniciamos o contador, desta vez de volta a cada 1s, colocando então o valor 10 de volta no ptc, e reiniciando a variável global para 0.

Para apresentar o resultado por 5 segundos, realizamos um loop onde o registo 0 tem o valor 5 e ficamos continuamente em loop a ler a variavel global que está a contar o tempo em segundos. Quando o tempo for igual a 5, volta ao inicio do programa.

5.

1.

Já realizado no tópico “Esquema de ligações”.

2.

Como nós queremos contar de segundo em segundo, colocamos a frequencia no ptc a 10hz e fazêmo-lo contar até 10 (0x000A). Como 10hz equivale a 1/10=0,1s, ele incrementa 1 bit no temporizador a cada 0,1s, e como ele tem que contar até 10, 10*0,1=1s, o que implica que ele entra no isr (interrupção) a cada segundo e assim incrementando a variável global que utilizamos para guardar o tempo.

Para calcular o estímulo precisamos de um tempo mais preciso. Neste caso, nós colocamos o ptc a contar até 1 (0x0001), o que implica que ele entra no isr a cada 0,1s e guardando assim na variavel global.

3.

O ptc recorre a 4 frequencias, 1hz, 10hz, 100hz e 1000hz. Como nós queremos a latência máxima, queremos também a frequência que tem maior tempo de execução. Neste caso seria o 1hz, pois equivale a adicionar 1 bit a cada (1/1=1) 1s. Como o ptc contem 8 bits de armazenamento, equivale a (255*1) = 255s de latência máxima.

4.

Para calculcar o tempo que a rotina demora, nós temos que calcular os ciclos que temos no isr. Neste caso, temos 11 instruções com 6 ciclos e 3 instruções com 3 ciclos. Agora como o p16 demora 20us (microsegundos) por cada ciclo podemos calcular o tempo que ele demora a executar o isr.

(11*6 + 3*3) * 20us = 1500us = 1,5ms = 0,0015s

❖Conclusão:

Com este trabalho, e com as diversas atividades laboratoriais ao longo do semestre, conseguimos perceber as diferentes possibilidades de ferramentas/programas que podemos criar, fazendo uso da linguagem assembly, bem como podemos acrescentar também funcionalidades externas usando, dispositivos exteriores ao sistema, como foi o caso específico deste trabalho, ao utilizarmos o pTC.