ISEL
Departamento de Engenharia Eletrónica e Telecomunicações e de Computadores

Licenciatura em Engenharia Informática e de Computadores
e
Licenciatura em Engenharia Informática Redes e Telecomunicações

Relatório do 4º Trabalho

*

Medição de Tempo de Reação

Trabalho realizado por:

Hélio Moura	Nº 49009
Romário Dias	Nº 50083
Iliano Santos	Nº 50096

Turma: 26D
Docente: Tiago Dias

Arquitetura de Computadores
2022 / 2023 verão

18 de junho de 2023

ÍNDICE

1 Introdução

O presente trabalho teve como principal objetivo explorar o uso de hardware em torno de processadores no desenvolvimento de programas em linguagem assembly. Os tópicos abordados incluem: portos paralelos de entrada e saída de dados, temporização, interrupções externas e organização de programas em rotinas e implementação de máquinas de estados em software.

Estes conceitos serão geridos em conjunto visando a construção de um projeto de arquitetura de computadores, nomeadamente um sistema de medição de tempo de reação.

Assimilar e utilizar de maneira efetiva esses recursos é fundamental para ampliar o desempenho e a eficiência dos programas. Ao explorar a interação entre software e hardware, este trabalho visa fornecer conhecimentos valiosos para a criação de programas otimizados e que possam interagir de maneira eficaz com o ambiente externo.

2 Arquitetura do protótipo

1ªQuestão:

Por forma a testar o nosso código produzido, começamos por estabelecer ligações para o desenvolvimento do sistema, onde utilizamos uma frequência de 1KHz para o CLK (no ATB), o sinal de nCS_EXT1 da placa SDP16 ao Chip Select do pTC. Decidimos usar a parte baixa da palavra de 16 bits dos registos do P16 (WRL), portanto utilizámos um barramento de dados dos bits 0 a 7 da palavra. Utilizámos os endereços A_1 e A_2 para fazer a ligação aos endereços A_0 e A_1 do pTC, respetivamente. Fizemos também a ligação dos sinais OE e nINT_EXT da placa SDP16 ao nRD e nINT, respetivamente.

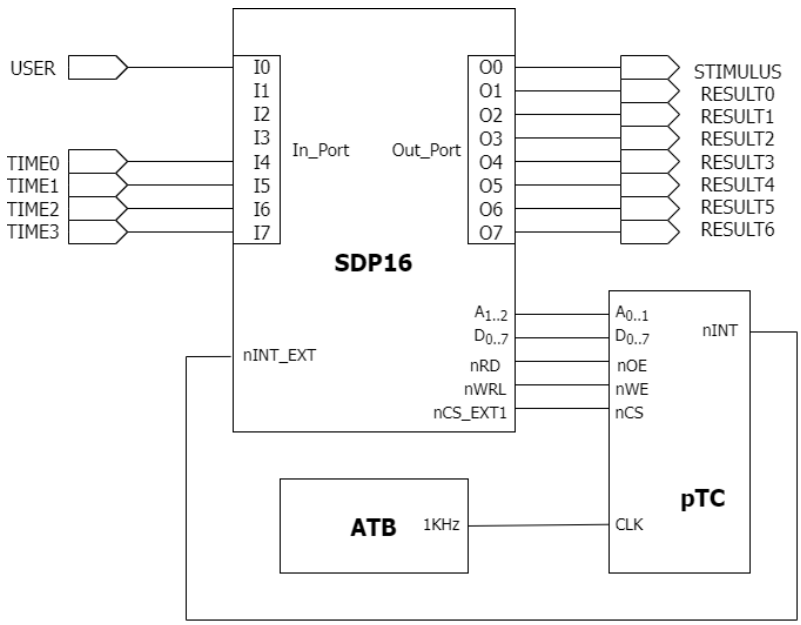

Figura 1: Esquema de ligações do sistema

No andamento da escrita do código, tivemos a necessidade de recorrer a um mapa de memória que se encontra descrito abaixo:

Figura 2: Mapa de memória do sistema

3 Funcionamento do sistema

1. No arranque do sistema o porto de saída tem o valor 0xFF, sabendo que a entrada USER está inativa (USER='0').

2. Após esta etapa de iniciação, quando USER passa a valor lógico '1' é realizado um teste de medição de tempo de reação, o que resulta na afixação do valor, o que implica dar reset à saída RESULT (0x00). Nesta etapa o usuário define o tempo de estímulo para o teste.

3. Na primeira etapa do teste, a saída STIMULUS mantem-se ativa durante o tempo de estímulo definido na etapa 2. Caso a entrada USER seja desativada antes dedo tempo de estímulo terminar, o teste é abortado e o sistema retorna à etapa 2. Quando o tempo de estímulo sem a desativação do USER, é guardado esse instante de tempo do pTC.

4. Na etapa seguinte do teste, a saída STIMULUS é desativada e o sistema fica a aguardar pela desativação da entrada USER, contabilizando o tempo que medeia entre ambos os eventos, fazendo a diferença entre este instante onde USER é desativado e o instante guardado na etapa 3.

5. Na última etapa do teste, o resultado do teste é afixado na saída RESULT por cinco segundos, definido pela rotina *delay*. Decorrido esse tempo, o sistema retorna à etapa 2.

2ªQuestão: Explique os cálculos realizados para determinar as temporizações envolvidas neste trabalho.
R: Escolhemos a frequência para o CLK de 1KHz, uma variável *sysclk* com o valor 9, indicando um novo pedido de interrupção a cada 10 ms.
T = $\frac{1KHz}{10}$ = 10 ms
Foi ainda definido o valor 5 para o DELAY.
3ªQuestão: Indique, justificando, a latência máxima do sistema no atendimento dos pedidos de interrupção gerados pelo circuito pTC.
R: A latência máxima do sistema no atendimento dos pedidos de interrupção gerados pelo circuito pTC é o valor definido pelo match register, ou seja, 10 milissegundos.

4ªQuestão: Indique, justificando, quanto tempo demora, no pior caso, a execução da rotina utilizada para o atendimento da interrupção externa.
R: Tendo em conta que são gerados pedidos de interrupção a cada 10 ms, a execução do programa ao sair da rotina '*isr*' e avalia se o estado mudou ou não. Caso o estado não tenha mudado, repete a execução da rotina e então fica aí e não avança, o que também acontece no pior caso quando há problemas no código que está sendo utilizado.

4 Conclusões

Após a entrega do código decidimos melhoramos ligeiramente a sua legibilidade, nomeadamente, removemos alguns comentários desnecessários e retiramos algumas constantes globais inutilizadas. Também acabamos por alterar o nome da rotina *do_test* para *condition_test*, visto que dentro dela é que necessariamente iriam ser avaliadas as condições para se realizar um novo teste de medição. Além de detetarmos algumas falhas no código que impediam o correto funcionamento do sistema, que infelizmente não os conseguimos resolver a tempo útil para entrega, mudanças que referem a execução da anterior rotina *do_test* (atual *condition_test*), onde verificamos que no momento em que ao avaliar a condição de teste, essa mesma condição fosse falsa, então o fluxo de execução repetia novamente a rotina, sem ter desempilhado, o que originava um constante empilhamento de valores para o stack (estava constantemente a fazer push), e portanto, para resolver isso definimos um ponto de execução a seguir ao empilhamento (execução de todos os '*push*') para que no caso de ser necessário voltar atrás na execução não seja empilhada novamente valores para o stack.

Removemos a rotina delay, removendo as suas chamadas, e acrescentando 2 pontos de execução *delay_seconds* e *show_result_by_delay_s*. Acresentamos uma variável global inicializada, que corresponde a um array, usado para processar os diferentes tempos, em milissegundos, para cada cada valor de TIME introduzido, assim como um *array_addr*. Removemos a utilização das rotinas *ptc_start* e *ptc_stop* pois do antigo *do_test*, pois a forma que estávamos a usar para codificar o sistema era desaconselhada. Alterou-se o *sysclk_get_ticks*, de forma que primeiro busca o endereço da variável *sysclk*, e a seguir busca o conteúdo desta variável.

Após realização do trabalho acreditamos ter alcançado a maior parte dos objetivos pretendidos, no entanto devido a alguma inexperiência demoramos um pouco mais a avançar nos requisitos do sistema. Por má interpretação, não implementamos no sistema um resultado totalmente correto, pois não se via o valor -64 no RESULT quando o TIME estivesse excedido. Isso seria resolvido calculando a diferença entre o tempo onde user foi colocado a '0' e o valor do tempo quando o bit stimulus foi a zero, ou seja, o tempo de estímulo terminou, e de seguida, verificar se esse valor da diferença teria ultrapassado o valor do TIME. No fim, acabamos por consolidar nossos conhecimentos nos conteúdos lecionados ao longo deste semestre letivo.