ISEL
Departamento de Engenharia Eletrónica e Telecomunicações e de Computadores

Licenciatura em Engenharia Informática e de Computadores

Arquitetura de Computadores
Relatório do Trabalho de Projeto
(Controlador de semáforos)

Trabalho realizado por:

Nome: Miguel Ângelo Campos Casimiro	Nº 51746
Nome: Jessé Alencar	Nº 51745
Nome: Cátia Abrantes	Nº 51747

LEIC22D – Grupo 09

Docente: Tiago Dias

2023 / 2024 Semestre de verão

6 de junho de 2024

Índice

1. INTRODUÇÃO 2

2. HARDWARE 3

2.1. CONFIGURAÇÕES DA PLACA SDP16 3

2.2. CONEXÕES ENTRE O PTC E A PLACA SDP16 4

2.3. TEMPORIZAÇÕES 4

3. SOFTWARE 6

3.1. MÁQUINA DE ESTADOS IDEALIZADA 6

3.2. DESCRIÇÃO DO CÓDIGO IMPLEMENTADO 6

4. CONCLUSÕES 9

5 BIBLIOGRAFIA 10

1. Introdução

Este relatório trata sobre o Projeto Final de Arquitetura de Computadores do semestre de verão do ano letivo 23/24, que teve como principal objetivo a exploração do hardware envolvente de um processador no desenvolvimento de programas escritos em linguagem assembly. Estiveram envolvidos os seguintes tópicos: entrada e saída de dados, temporização, interrupções externas, organização de programas em rotinas e implementação de máquinas de estados em software.

2. Hardware

A placa SDP16[1] é um sistema didático para exploração do processador P16, servindo de apoio ao estudo da linguagem *assembly* P16, dos mecanismos de endereçamento utilizados pelo mesmo no acesso aos dispositivos de memória e aos periféricos.

A placa possui uma memória RAM de 32 KB, acessíveis à palavra (16 bits) e ao byte (8 bits), e dois portos de entrada de 8 bits, um de entrada e outro de saída, para interação com o exterior.

2.1. Configurações da placa SDP16

Para a realização do Projeto foi necessário o uso dos portos de entrada e saída presentes na placa SDP16 de modo a interagir com o porto de entrada, ou seja, o botão de pedestres e os DIP-*switches* presentes na placa para que o programa pudesse obter informações, processá-las e reagir de acordo, enviando informação através do porto de saída para os LED's presentes no módulo LAPI.

Os periféricos associados aos portos de entrada e saída da placa que foram utilizados neste projeto são os seguintes:

- O botão de pressão PB1 do módulo de expansão LAPI, utilizado para simular os pedidos de atravessamento dos peões foi associado ao bit 0 do porto de entrada;
- O interruptor 4 do DIP-*switch* SW1 instalado na placa SDP16 está associado ao bit 4 do porto de entrada e foi utilizado para estabelecer o valor do sinal CONFIG, que quando está a nível '1' lógico coloca o sistema no modo de configuração;
- Os interruptores 5 a 7 do DIP-*switch* SW1 estão associados aos bits 5 a 7 do porto de entrada e foram utilizados para estabelecer o valor do sinal TIME, que é utilizado para definir o tempo que o sinal verde de travessia de peões deve estar aberto;
- Os bits 0 e 1 do porto de saída estão associados ao LED 1 do módulo LAPI, e cada bit controla uma cor do LED, a vermelha e a verde, respectivamente;
- Os bits 2 e 3 do porto de saída estão associados ao LED 2 do módulo LAPI, controlando as cores vermelha e verde, respectivamente;
- Os bits 3 e 4 controlam as cores vermelha e verde do LED 3, presente no módulo LAPI, respectivamente.

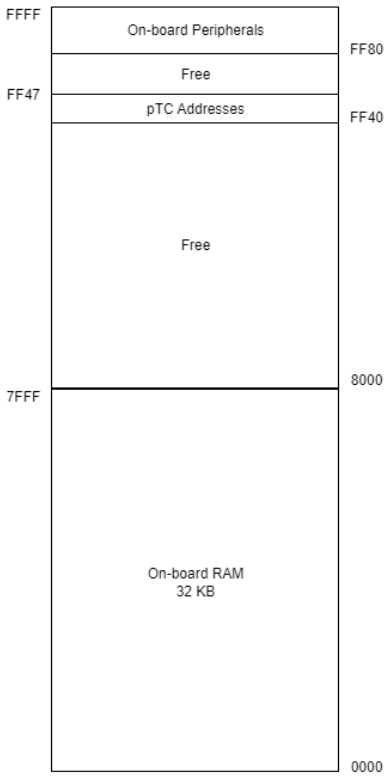

Figura 1 – Mapa de Endereçamentos do Sistema

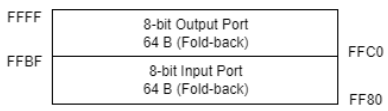

Figura 2 – Mapa de Endereçamento dos Portos de Entrada e Saída

[1] Para informações mais aprofundadas consultar o Manual de Utilização da placa SDP16

2.2. Conexões entre o pTC e a placa SDP16

A placa SDP16 disponibiliza dois sinais de seleção (nCS_EXT0 e nCS_EXT1) para duas zonas do mapa de endereçamento, cada uma com dimensão de 64 B. O sinal EXT0 está associado à gama de endereços 0xFF00 a 0xFF3F, enquanto o sinal EXT1 está associado à gama de endereços 0xFF40 a 0xFF7F.

Para o uso do pTC foi escolhido o sinal EXT1 como sinal de seleção, pelo que o mesmo está contido na gama de endereços 0xFF40 a 0xFF7F. O pTC possui quatro registos internos, acessíveis ao byte, que decidimos associar aos primeiros 8 bytes de endereço onde o sinal EXT1 está disponível, como pode ser visto na Figura 3.

Figura 3 – Mapa de Endereçamento do pTC

Uma vez escolhido o sinal de seleção e os endereços nos quais o dispositivo é acessível, fizemos as seguintes conexões[2] entre a placa SDP16 e os pinos de entrada do pTC:

- Uma vez definidos os endereços onde o pTC é acessível, o sinal de seleção do pTC, nCS, está ligado ao sinal nCS_EXT1 da placa;
- Os pinos de endereço $A_{0..1}$ do pTC estão ligados ao barramento de endereços da placa nos bits $A_{1..2}$;
- Os pinos de dados $D_{0..7}$ do pTC estão ligados ao barramento de dados da placa nos bits $D_{0..7}$, ou seja, à parte baixa da palavra, de modo a aceder aos registos do pTC nos endereços pares;
- O pino que permite a leitura de dados, nOE, está ligado ao sinal nRD, da placa;
- Uma vez que os registos estão acessíveis na parte baixa da palavra, o pino que permite a escrita de dados, nWE, está ligado ao sinal nWRL.

2.3. Temporizações

Durante o desenvolvimento do sistema foi necessária a implementação de um relógio de sistema, para que fosse possível contabilizar a passagem do tempo, uma parte essencial do Projeto, já que permitiria a alteração do estado dos LED's (ligado/desligado) num intervalo padrão e bem definido. Para isso foram escolhidas a frequência de clock aplicada no pTC e o valor limite de contagem do mesmo.

Uma vez que em ambos os (ISEL, Maio de 2024)estados deveriam existir LED's a piscar ao ritmo de 0.5 segundos, foi necessário definir uma unidade padrão para o relógio do sistema que fosse menor do que 0.5 segundos e que trabalhasse com valores inteiros, e não com vírgulas, pelo que decidimos que a unidade do relógio do sistema seria equivalente a 0.1 segundos (1 décimo de segundo). Com a unidade de medição definida, era preciso identificar uma frequência de clock e um valor limite para o pTC, que ao atingi-lo, ativaria o sinal de interrupção externa do sistema para que este incrementasse em 1 o relógio do sistema.

[2] Para informações sobre onde as conexões físicas foram feitas, basta consultar o Capítulo 5 do Manual de Utilização da placa SDP16 e a configuração de pinos do pTC, presente no pTC_Datasheet.

Existem múltiplas maneiras de implementar o relógio do sistema, mas tendo observado as frequências disponíveis na placa ATB, que são 1 Hz, 10 Hz, 1 kHz e 10 kHz, decidimos utilizar a frequência de 1 kHz como a frequência de clock do pTC, pelo que o pino de clock deste circuito foi ligado ao sinal de 1 kHz da placa ATB. Tendo a frequência de clock definida, aplicamos a fórmula:

$$\frac{f_{clock}}{valor\ limite\ do\ pTC+1} = f_{incremento\ do\ sysclk} \Leftrightarrow \frac{valor\ limite\ do\ pTC+1}{f_{clock}} = 1\ sysclk$$

$$\frac{match+1}{1000} = 0{,}1 \Leftrightarrow match + 1 = 100 \Leftrightarrow match = 99$$

Logo temos que o valor que deve estar contido no Terminal Match Register do pTC é 99, pois o contador inicia a contagem em 0 (por isso *"match + 1"*).

Com esses valores definidos foi possível programar o pTC para contar até 99 (100 ticks), de modo a obter o relógio do sistema cuja unidade corresponde a 0.1 segundos.

3. Software

3.1. Máquina de estados idealizada

Antes de começarmos a escrever código assembly, implementamos uma máquina de estados UML (Figura 2), o que permitiu uma maior facilidade no desenvolvimento e compreensão da aplicação desenvolvida.

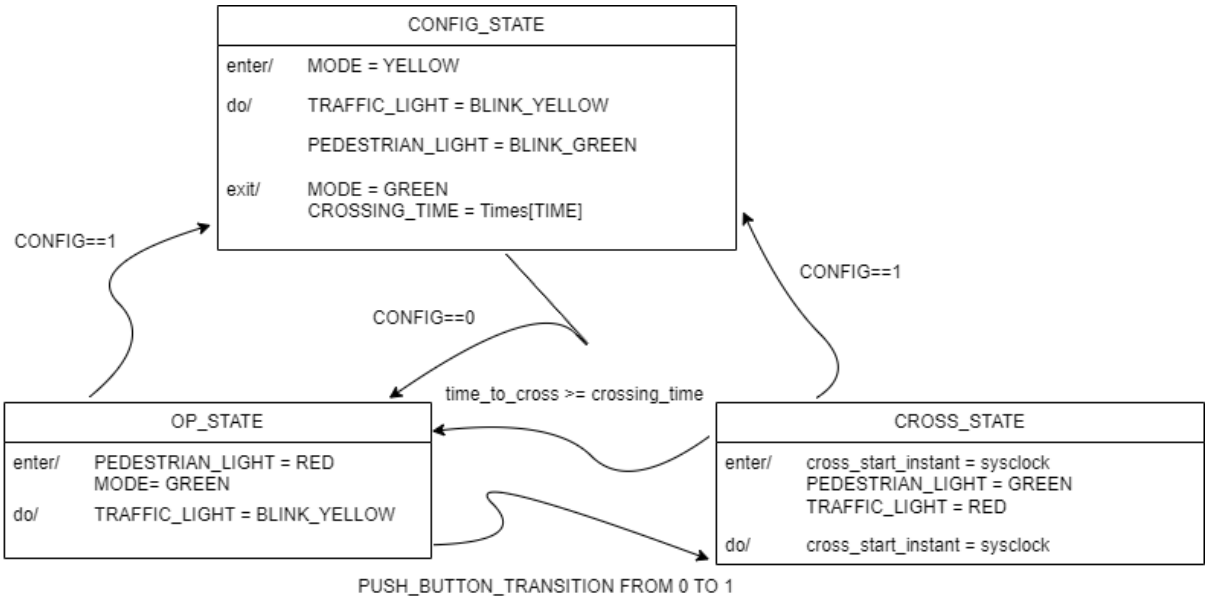

Figura 4 – Máquina de estado UML

3.2. Descrição do código implementado

Relativamente ao código utilizado é possível afirmar que foram utilizadas rotinas e subrotinas ao nível de software, hardware e middleware, na implementação do gestor de periféricos, também no que consta à temporização e contagem do periférico PTC explorada pela rotina sysclock através da interrupção externa e por fim o código assembly ao nível da aplicação.

A rotina **op_state:** inicializa os LED's do modo operação e realiza a leitura do porto de entrada para verificar se o switch de configuração foi alterado ou se o botão dos pedestres foi pressionado, ainda dentro dessa rotina, existem mais três subrotinas, a subrotina op_state_do: que é um loop que verifica se o switch de configuração foi alterado ou se o butão dos pedestres foi pressionado, a subrotina **lights_on_off**: para a mudança de estados dos LED's e a subrotina **lights_on:** para a ativação dos LED's. Além disso a rotina **p_buton:** verifica se houve uma transição no botão dos pedestres que é complementarda pela subrotina **no_transaction:** que caso não ocorra uma transição é devolvido o valor anterior do botão. Também foi usada a rotina **isr:** que permite ao processador atender às interrupções externas já implementas nos laboratórios ao longo deste semestre.

A rotina **to_config_button**: verifica o botão de configuração no estado de operação. Caso este contenha o valor lógico '1', então o programa muda para o estado de configuração, caso contrário retorna à rotina onde foi chamada. Enquanto que a rotina **from_config_button**: verifica o botão de configuração no estado

de configuração. Caso este contenha o valor lógico ‘0’, então o programa muda para o estado de operação, caso contrário retorna à rotina onde foi chamada.

Na rotina de configuração, o programa irá acender todos os LED’s necessários, LED do modo a amarelo, LED dos carros a amarelo e LED dos pedestres a verde. De seguida, marca-se o tempo em que os LED’s foram acesos, sendo este mesmo valor guardado na memória, passando a ser o tempo anterior.

Guarda-se também na memória o valor lógico ‘1’, indicando o estado dos LED’s (‘1’- LED’s acesos & ‘0’- LED’s apagados), passando, de seguida para a subrotina **config_state_do**: onde se irão observar os três bits de maior peso do porto de entrada que irão indicar a posição de um array de tempos de espera, deste modo poderá ser alterado, durante o estado de configuração, o tempo do atravessamento da faixa de rodagem. Para além disto, durante esta subrotina, irá também ser verificado durante quanto tempo os LED’s se mantiveram acesos/apagados. Para isso, irá ser verificado o tempo atual e este será comparado com o tempo anterior que será lido da memória, verificando também o estado do botão de configuração. Caso o tempo calculado seja maior ou igual ao tempo requerido (0,5s), então irá passar para outra subrotina (**config_lights_on_off**), caso contrário irá voltar ao início desta sub-rotina para volta a verificar tudo.

Na subrotina **config_lights_on_off**: será verificado o estado dos LED’s, se estes estiverem acesos então irão ser apagados e será guardado na memória um novo ‘tempo anterior’ correspondente ao momento em que os LED’s foram apagados. Se os LED’s estiverem acesos irá para passar para outra sub-rotina onde estes serão apagados e será guardado na memória um novo ‘tempo anterior’ correspondente ao momento em que os LED’s foram acesos. Em ambos os casos, depois destas instruções o programa irá voltar para a sub-rotina ‘config_state_do’, para voltar a verificar tudo.

Por último temos a rotina **cross_state**: irão ser ligados os LED’s necessários, LED do modo a verde, LED dos pedestres a verde e LED dos carros a vermelho. Irá ser registado e guardado na memória o momento em que os LED’s foram acesos para que seja possível o cálculo do tempo de atravessamento da faixa de rodagem, passando, de seguida para a subrotina **time_to_cross**: onde será verificado o tempo atual e carregado o tempo de atravessamento da faixa de rodagem, sendo também verificados os botões de configuração e pedestres. A partir da subtração entre o tempo atual e o anterior irá ser verificado se já foi atingido o tempo requerido (tempo de atravessamento da faixa de rodagem). Caso isto aconteça irá voltar de novo para a rotina **op_state**, caso contrário irá voltar a **time_to_cross**.

Quanto à rotina de interrupção, quando o sinal de interrupção nINT é ativado, o Program Counter recebe o valor 0x0002, que corresponde a uma instrução de load para o PC do endereço da rotina de interrupção, pelo que levam 6 ciclos de relógio, ou seja, 120 microssegundos para o início do atendimento do pedido de interrupção gerado pelo pTC. A rotina de interrrupção possui 14 instruções, sendo 10 de acesso a memória, seja para realização de load’s e store’s ou para push’s e pop’s, e 4 instruções normais, que não acedem à memória, logo a rotina demora 72 ciclos de relógio, ou seja, 1.44 milissegundos para atender ao pedido de interrupção e incrementar a variável sysclk. Uma vez que ocorre um pedido de interupção a cada 100 milissegundos e a rotina de interrupção leva no total 1.56 ms para ser realizada, restam 98.44 milissegundos para que as restantes execuções do programa possam ser executadas. No pior caso, considerando que todas as instruções restantes são de acesso à memória, ou seja, que levam 6 ciclos de relógio para serem

executadas, podem ser executadas 820 instruções entre um pedido de interrupção e o próximo. Segue abaixo os cálculos que comprovam essa quantidade de instruções entre os pedidos:

$$1\ ciclo\ de\ relógio = 2 * 10^{-5} s\ ; \qquad tempo\ restante\ para\ instruções = 98{,}44 * 10^{-3} s$$

$$\frac{tempo\ restante\ para\ instruções}{6\ ciclos\ de\ relógio} = \frac{98{,}44 * 10^{-3}}{12 * 10^{-5}} = 820{,}333(3) \cong 820\ instruções$$

4. Conclusões

A realização deste projeto final permitiu-nos desenvolver um maior conhecimento acerca da interação com os portos de entrada e saída do processador, tanto como a interação entre o hardware e software utilizados. Relativamente ao software foram exploradas a implementação de estados, a organização de rotinas e a utilização de pedidos e rotinas de interrupção de modo a com um relógio de sistema, o que resultou no aprimoramento das nossas habilidades práticas relativamente à implementação de rotinas assembly. Na realização deste trabalho a maioria dos objetivos foram completados, sendo a única exceção a extensão do tempo de atravessamento da faixa de rodagem após uma nova transição ascendente do botão de pedestres, estando o sistema no estado de atravessamento.

5 Bibliografia

Dias, Tiago: *Pico Timer/Counter (pTC) - Product Datasheet.* ISEL - IPL, Lisboa, Portugal. (junho 2021).

ISEL: *Controlador de semáforos (v1.1) - Trabalho de Projeto - Arquitetura de Computadores.* ISEL, Lisboa, Portugal. (Maio de 2024).

Paraíso, José, & Dias, Tiago: *Placa de Desenvolvimento SDP16 - Manual de Utilização.* ISEL, Lisboa, Portugal. (março 2023).

Sampaio, Pedro: *SDP16 - LED & Push Button Interface - Schematics.* ISEL - IPL, Lisboa. (maio 2024).