Instituto Superior de Engenharia de Lisboa

ARQUITETURA DE COMPUTADORES

Relatório do Terceiro Trabalho Prático

Docente

Prof. Hernâni Mergulhão

Alunos

43552 Samuel Costa
43884 Pedro Rocha

Julho 2017

Índice

Introdução

Este relatório, referente ao terceiro de três trabalhos práticos, foi elaborado no âmbito da unidade curricular de Arquitetura de Computadores e acompanha a implementação de um semáforo para gestão do fluxo de pessoas numa passagem bi-direcional estreita, com base no sistema PDS16.

Em cada uma das extremidades, o sistema é composto por um botão que serve para pedido de autorização de passagem, por dois sensores IV (simulados por switches na placa de teste ATB), que permitem contabilizar os visitantes em trânsito e dois avisadores luminosos, que indicam se a é permitida ou não a passagem.

Ao porto de entrada do PDS16 foram ligados os dois botões e quatro sensores; no porto de saída fizeram-se representar os 4 avisadores luminosos. Foi também utilizado o temporizador SASO TIMER v.3, para cumprir as temporizações especificadas.

Ao longo do documento apresentam-se os recursos utilizados; uma explicação dos procedimentos aplicados na execução das tarefas propostas, incluindo a elaboração de um programa em linguagem assembly do PDS16 que representa uma solução ao problema proposto, e elaboração de um programa de teste que permitiu responder às perguntas postas pelo enunciado do trabalho e aprofundar os conteúdos em estudo.

Métodos e Recursos Utilizados

Para a realização deste trabalho foi utilizada a aplicação DASM, PDS16, placa de teste ATB, PDDebugger v2.0 e PDSimulator v1.2.

Trabalho Realizado

Aplicação

Conceção e realização

Foi concebida e desenvolvida uma aplicação em linguagem *assembly* do PDS16, que, depois das inicializações necessárias, adquire uma imagem das entradas, decompondo-a nos seus diferentes elementos, e tornando-a coerente para tratamento pelo controlo.

Assim, esta aplicação é composta por duas secções. Em primeiro lugar, são efetuadas inicializações: os avisadores luminosos são desligados; são iniciadas a zero as variáveis a serem avaliadas pelo controlo. Ao proceder assim, assumiu-se que o botão RESET só é pressionado depois de se ter garantido que a passagem se encontra vazia, e que eventuais pedidos de passagem efetuados anteriormente devem ser ignorados. É *enviado* o valor correspondente a um segundo para o timer, e é ativada a flag *Interrupt Enable,* dando permissão à aplicação para atendimento de interrupção, assim que ocorrer um pedido.

A rotina que configura o temporizador e a rotina que ativa o sinal IE do registo PSW demoram 24 e 10 impulsos de relógio a serem executadas.

Em seguida, é executado um ciclo(loop), composto pelas três rotinas Input_Data, que obtem uma imagem do porto de entrada, guardando-a em memória direta. ParseInput, que efetua a decomposição da entrada de 8bits conforme correspondência na figura xx e Accounting, que dado o estado atual e anterior dos sensores de presença, contabiliza entradas e saídas.

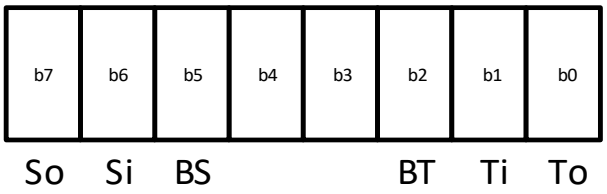

Figura 1- Correspondência entre pesos dos bits do porto de entrada e os sinais do sistema semáforoo

O ciclo demora, no pior caso, 462 impulsos de relógio a ser executada, conforme a tabela na Figura 2. Assim, o programa demora 462 + 34 = 496 impulsos de relógio a realizar as rotinas referidas. Como a frequência do SDP16 é 500Hz, o programa demora $\frac{1}{500} x496 = 0{,}992s$ a disponibilizar a informação "mais atualizada" a ser consumida pela rotina de interrupção. Como $0{,}992s$<1s, podemos concluir que quando ocorrer o primeiro pedido de atendimento de interrupção, os dados a serem consumidos pela rotina que atende o pedido correspondem ao estado mais recente da máquina de estados.

Secção	Contagem de impulsos
main (init)	92 (58+24+10)
loop:	462 (30+18+28+42+42+20+42+120+120)

Figura 2 – Contagem de impulsos de relógio para as as duas secções da aplicação

A rotina de atendimento de interrupção inclui parametrização de quando ocorrerá o próximo pedido de atendimento de interrupção. Assim, qualquer pedido de atendimento de interrupção é processado tão logo quanto ocorra.

Teste no PDS16

A aplicação desenvolvida foi testada no PDS16. Inicialmente, foi usado o oscilador de 1Hz disponibilizado na ATB e enviado o valor 1 para porto de saída associado à gama de endereços NCS_EXT0, a cujo porto foram ligadas as entradas do timer. Foi usado o PDDebugger, para carregar o programa desenvolvido em memória. Foi escrita uma rotina de atendimento de interrupção para testar o funcionamento da aplicação, publicando no porto de saída o valor da variável number of People. Também foi testado o funcionamento dos botões desenhando uma rotina de atendimento de interrupção que mostrasse o valor actual dos botões no porto de saída do PDS16.

Rotina de Atendimento de Interrupção

Para realizar o controlo do sistema, foi projetada uma máquina de estados, conforme diagrama da figura.

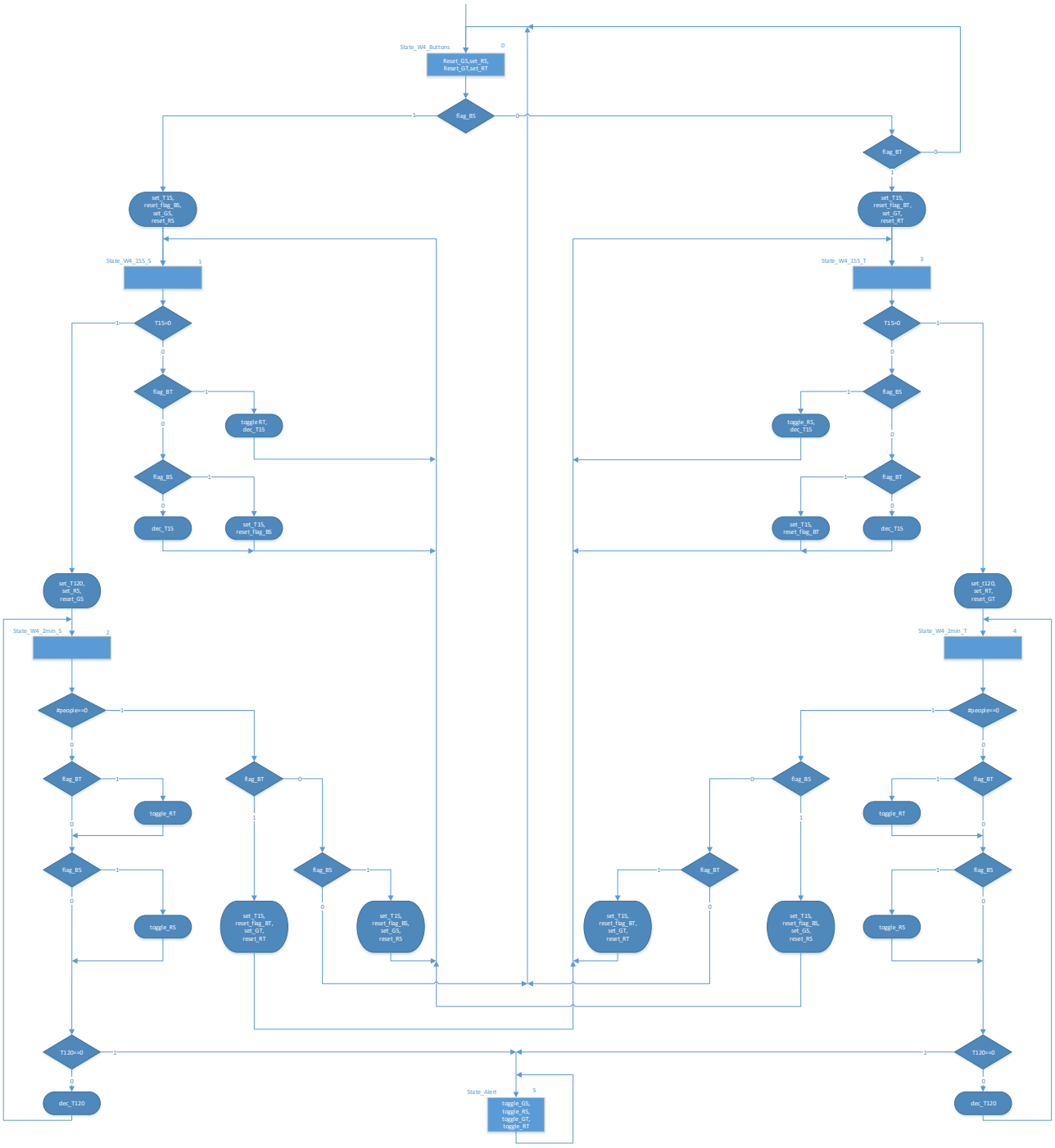

Figura 3 – Diagrama ASM da máquina de controlo implementada na rotina de atendimento de interrupção

Gestão dos avisadores

Temporizações

Para obter as temporizações pedidas no enunciado e minimizar o erro associado foi escrito um programa que implementa um contador *UP*. Quando o bit de peso 11 do contador for 1 (Count >0x800) é acendido o LED de menor peso do porto de saída. A gestão da saída é realizada por uma rotina de interrupção, que mantém um contador *local*. O programa foi executado e foi registado o tempo até até O0 acender. Consultou-se o valor presente no contador da rotina de atendimento de interrupção.

Assim, calculou-se a frequência do CPU, dividindo o número de ciclos que o programa demora a executar, pelo tempo de execução. O total de ciclos executados pelo processador é dado por #Cycles_main + (#Cycles_ISR_not_yet * Count_ISR-1)+ #Cycles_ISR_publish, em que:

- #Cycles_main é o número de ciclos que a rotina principal demora a executar até se obter o output pretendido;

-#Cycles_ISR_not_yet – número de ciclos que a rotina de atendimento toma quando ainda não há lugar a publicação do resultado no porto de saída;

- Count_ISR é o valor consultado na word Count_ISR depois de se ter terminado a execução do programa;

- #Cycles_ISR_publish – número de ciclos que a rotina de atendimento de interrupção demora a executar quando há lugar a publicar o resultado no porto de saída.

Assim, o total de ciclos executados pelo processor é $41042 + (48x73) + 64 = 44610$. Como o programa demorou 67seg a acender o bit de maior peso do porto de saída, temos a frequência real de funcionamento é $\frac{44610}{67} = 665{,}821$ Hz.

Foi utilizado o oscilador 10Hz da ATB. O temporizador foi carregado com o valor 10. Foi registado o tempo de 67seg. Foram realizados 73 atendimentos de interrupção, pelo que o erro introduzido pelo temporizador utilizado é $\frac{73-67}{67} = 8{,}9\%$. Foram introduzidos outros valores no temporizador, por forma a diminuir o erro. Diminuiu-se uma unidade o valor carregado no temporizador, o erro obtido foi $\frac{83*0{,}9-66}{68} = 12{,}79\%$. Por outro lado, aumentando o valor em uma unidade obteve-se $\frac{69*0{,}9-66}{66} = 5{,}9\%$ de erro. Considerou-se que este era o valor óptimo para o hardware envolvido.

As duas temporizações relevantes no âmbito das decisões tomadas pela máquina de estado foram testadas, tendo-se obtido os seguintes resultados.

Foi testada a aplicação desenvolvida introduzindo na rotina init_timer o valor 11 para o temporizador, tendo-se registado 15,68s para T15 e 118,25s para T120, representando erros na ordem dos 4,5% e 1,46%.

Decorrem aproximadamente 15,68s desde que um pedido é atendido até que, sem ter ocorrido novo pedido, o sistema esteja de novo pronto para aceitar um pedido. Tal encontra-se em linha com o esperado. Decorrem aproximadamente 118,25s desde que ocorre a primeira transição para o estado 2 (ou 4), até que é ativado o alarme silencioso. O resultado obtido corresponde a um erro de aproximadamente 1,46%.

Para testar se a extensão do programa desenvolvido e o seu tempo de execução punham em causa a possibilidade de obter uma imagem atualizada dos botões e sensores, foi usado o programa PDDebugger, para carregar no PDS16.

Foram introduzidos *breakpoints* na última linha da rotina loop (instrução jmp loop), e na linha 2 do programa (a posição de memória que PC toma quando ocorre interrupção). Constatou-se que a

primeira instrução é executada antes da segunda. Retirou-se, então, o primeiro *breakpoint*, e fez-se *RESET* ao processador, tendo-se observado que, quando o primeiro pedido de interrupção é atendido, R5 (link) tem o valor 0x00BA. Ou seja, quando é iniciada, a aplicação executa uma vez o ciclo loop completo, sendo ainda possível obter uma segunda imagem do exterior antes de ocorrer um pedido de atendimento de interrupção.

No pior caso a execução da rotina utilizada para o atendimento da interrupção demora 226 impulsos de relógio. Ou seja, temos que essa rotina demora, no pior caso, $226\ x\ \frac{1}{665{,}821} = 0{,}340\ s$ a ser executada.

Intermitência

Foi obtido o efeito de intermitência nos estados em que tal foi necessário, reservando o registo r2 para assinalar os bits em que foi necessário fazer toggle. Antes de publicar os avisadores, realiza-se uma união exclusiva entre o valor dos avisadores a publicar e a informação relativa aos bits a fazer intermitência (*toggle*).

Referências

[1] Esquemas elétricos – SPD 16 disponível em http://pwp.net.ipl.pt/cc.isel/ezeq/arquitetura/sistemas_didaticos/pds16/hardware/SDP16-Sch.pdf acedido a 13/07/2017

[2] SASO TIMER v3 datasheet – disponível em https://adeetc.thothapp.com/classes/AC/1617v/LI21N/workitems/2089/attachment acedido a 13/07/2017