Microprocessadores

BCC - UNESP/RC

Relatório do Trabalho Final da Disciplina de Microprocessadores

Docente: Alexandro José Baldassin

Pedro Bruce de Lima

Vitor Ferreira

1. Introdução

O seguinte relatório trata do desenvolvimento do projeto final da disciplina de Microprocessadores. O projeto tem como objetivo a aplicação dos conceitos aprendidos em aula, como comunicação serial, I/O, interrupções, temporizador, manipulação de bits e subrotinas.

Neste projeto, desenvolvemos um aplicativo console que aceita comandos enviados pelo usuário e, para cada comando, uma ação deve ser executada na placa DE2 Altera.

O comando 00 xx acende o xx-ésimo led vermelho, enquanto o comando 01 xx apaga o xx-ésimo led. O comando 10 inicia a animação dos leds, acendendo da esquerda para a direita ou da direita para a esquerda conforme o estado da chave SW0. Os leds são acesos e apagados a cada 200ms. O comando 11 pausa a animação dos leds. O comando 20 inicia o cronômetro de segundos utilizando 4 displays de 7 segmentos. O cronômetro pode ser pausado e resumido ao pressionar o botão KEY1. O comando 21 interrompe o cronômetro.

2. Desenvolvimento

2.1. Cronograma Inicial

Data	Atividade
17/10	Definir cronograma
14/11	UART e acionar leds
21/11	Interrupção e animação
28/11	Animação e cronômetro
05/12	Cronômetro

2.2. 14/11

Foi desenvolvida a função *main.s* capaz de: ler e escrever entradas do teclado no terminal; salva o último caractere digitado; SE for digitado um ENTER, compreende que um comando foi inserido e desvia para seu tratamento.

A sub-rotina *trata_led.s* distingue se foi comandado para acender ou apagar um LED, e também qual LED o comando referencia. Foi implementado o código para acender um determinado LED, porém ao repetir o comando com outro alvo o LED anteriormente aceso se apaga.

2.3. 21/11

Seguindo o desenvolvimento de *trata_led.s*, foi feita uma forma de salvar o contexto dos leds anterior ao comando, ou seja, leds anteriormente acesos assim permanecem até um comando apagá-los. Também foi implementada a função de apagar LEDs utilizando a ideia de obter o inverso da máscara usada para acendê-los.

Prosseguindo para o desenvolvimento da animação dos LEDs, foi configurado o temporizador para um período de 200 ms, encerramos o dia trabalhando com interrupções e tentando fazê-las funcionar conforme comandos do teclado.

2.4. 28/11

Prosseguimos com a implementação correta de interrupções conforme comandos do teclado e, após diversas tentativas, chegamos a conclusão de que seria necessário a criação de *stack frames* ao longo das interrupções para garantir que não estivéssemos utilizando o mesmo registrador mais de uma vez.

Após a conclusão dessa etapa, prosseguimos para a criação do *trata_anima.s*, rotina que será responsável por tratar a animação dos leds da placa. Configuramos a rotina e conseguimos executá-la com êxito. Ela é capaz de, ao ser interrompida com o comando 11 e retomada com 10, retornar ao estado anterior do último led que estava aceso antes da animação ser interrompida e continuar a animação a partir dele. Também configuramos a utilização da chave SW0 para, a depender do estado dela, a animação acontecer da direita para a esquerda ou da esquerda para a direita. Foi definido que, se ela estiver para baixo, a animação acontecerá da esquerda para a direita. O código irá pular para uma label onde o endereço do primeiro led da esquerda será carregado e os leds seguintes vão sendo acesos conforme um comando *srli*. Caso a chave esteja para cima, o código irá pular para uma label onde o endereço do primeiro led da direita será carregado e os leds seguintes vão sendo acesos conforme um comando *slli*. Dentro dessas labels, há uma comparação em cada uma que verifica se todos os leds já foram acesos e, caso seja verdadeiro, os endereços são carregados novamente.

Foi definido que, caso haja leds acesos e a animação seja ativada, todos os leds serão apagados para dar início à animação e os estados dos mesmos não são recuperados após pausar a animação. Concluímos a criação do *trata_led.s* e encerramos o dia.

2.5. 03/12

Este dia não estava previsto no cronograma inicial, porém como

o mesmo estava um pouco atrasado, achamos melhor comparecer ao laboratório para dar continuidade no projeto. Trabalhamos na criação do cronômetro com *crono.s* e conseguimos realizar uma implementação inicial do mesmo, sendo possível iniciá-lo com o comando 20 e contar até 99.

2.6. 04/12

Comparecemos no laboratório neste dia para darmos continuidade no desenvolvimento de *crono.s*. A rotina agora é capaz de tratar a casa das centenas e milhares, sendo possível haver uma contagem até 9999. Seguimos com o desenvolvimento da implementação da pausa e resumo do cronômetro com o botão KEY1.

O botão KEY1 foi configurado para pausar e resumir o cronômetro. Foi setado uma flag para verificar se o cronômetro foi pausado ou não. Para cada unidade de medida do cronômetro (unidade, dezena e centena) foi implementado um contador e um ponteiro para os códigos dos 7-SEG display equivalentes aos números decimais. O contador serve para saber quando converter o valor (10 unidades = 1 dezena, 10 dezenas = 1 centena, 10 ecntenas = 1 milhar) e os ponteiros servem para incrementar cada display de acordo (HEX0 = unidades, HEX1 = dezenas, HEX2 = centenas, HEX3 = milhares). A casa dos milhares é a única não capaz de "resetar" na implementação, ou seja, quando 10 milhares são contados, não há conversão e o display HEX3 não retorna a zero.

3. Conclusão

O projeto desenvolvido foi capaz de abordar grande parte dos assuntos vistos em sala de aula. Foi possível concluir com êxito tudo que foi proposto no enunciado. O projeto possui duas pequenas limitações apenas: o cronômetro não zera ao chegar em 9999 e, na animação da esquerda para a direita, o led LEDR0 não acende. Não tratamos dessas limitações por receio de perder muito tempo no desenvolvimento do projeto e acabar não entregando tudo o que foi proposto.