UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
INSTITUTO DE GEOCIÊNCIAS E CIÊNCIAS EXATAS

Departamento de Estatística, Matemática Aplicada e Computação

Bacharelado em Ciência da Computação - Noturno

RELATÓRIO DO PROJETO DE MICROPROCESSADORES

MAIRA BERLATO FIDALGO

GUILHERME HENRIQUE ZAMPRONIO

PEDRO HENRIQUE POTENZA FERNANDES

RIO CLARO
2023

Sumário

1. Introdução

Neste relatório abordaremos o funcionamento de nosso programa, ou seja, as questões que ficaram em abertos nas instruções do projeto e como o nosso código se comporta nessas situações. Também colocaremos um breve resumo de cada um dos arquivos e suas respectivas funções presente em nosso trabalho, assim como o seu funcionamento e as estratégias utilizadas durante o desenvolvimento.

Nesse projeto utilizamos a linguagem de montagem do Nios II e os recursos da DE2 Media Computer para desenvolver uma aplicação que abrangesse todas as funcionalidades e particularidades estudadas nos laboratórios de aula. O uso da DE2 Media Computer se deve principalmente ao fato de termos a disposição a operação div que foi utilizada como estratégia em algumas funções em nosso código.

Por fim, durante a conclusão abordaremos os desafios e problemas que foram enfrentados e a maneira os quais eles foram superados, assim como as lições e aprendizados obtidos durante o desenvolvimento.

2. Comportamentos

Tabela de comandos do sistema:

Comando	Ação
00xx	Piscar xx-ésimo led vermelho em intervalos de 500ms.
01xx	Cancelar a piscagem do xx-ésimo led vermelho.
10	Ler o conteúdo das chaves (8 bits – SW7-SW0) e calcular o respectivo número triangular. O resultado deve ser mostrado nos displays de 7 segmentos em decimal.
20	Inicia cronômetro de segundos, utilizando 4 displays de 7 segmentos. Adicionalmente, o botão KEY1 deve controlar a pausa do cronômetro: se contagem em andamento, deve ser pausada; se pausada, a contagem deve ser resumida.
21	Cancela cronômetro

O sistema define três principais funcionalidades: piscagem de LEDs, cronômetro e cálculo de número triangular. As funcionalidades do cronômetro e do número triangular não são compatíveis, ou seja, o sistema não é capaz de exibir o valor triangular ao mesmo tempo que é exibido o timer do cronômetro, se o cronômetro está rodando e é acionado a função triangular, o resultado é exibido rapidamente no display e em seguida o display volta a exibir o cronômetro rodando. As demais funções podem ser utilizadas em conjunto.

Há um erro de comportamento no sistema quando o primeiro comando a ser digitado é **20**, que não foi solucionado por ser uma situação muito específica e a dificuldade na compreensão em relação ao que estava falhando, uma vez que usando breakpoints, o erro não ocorria.

3. Flags

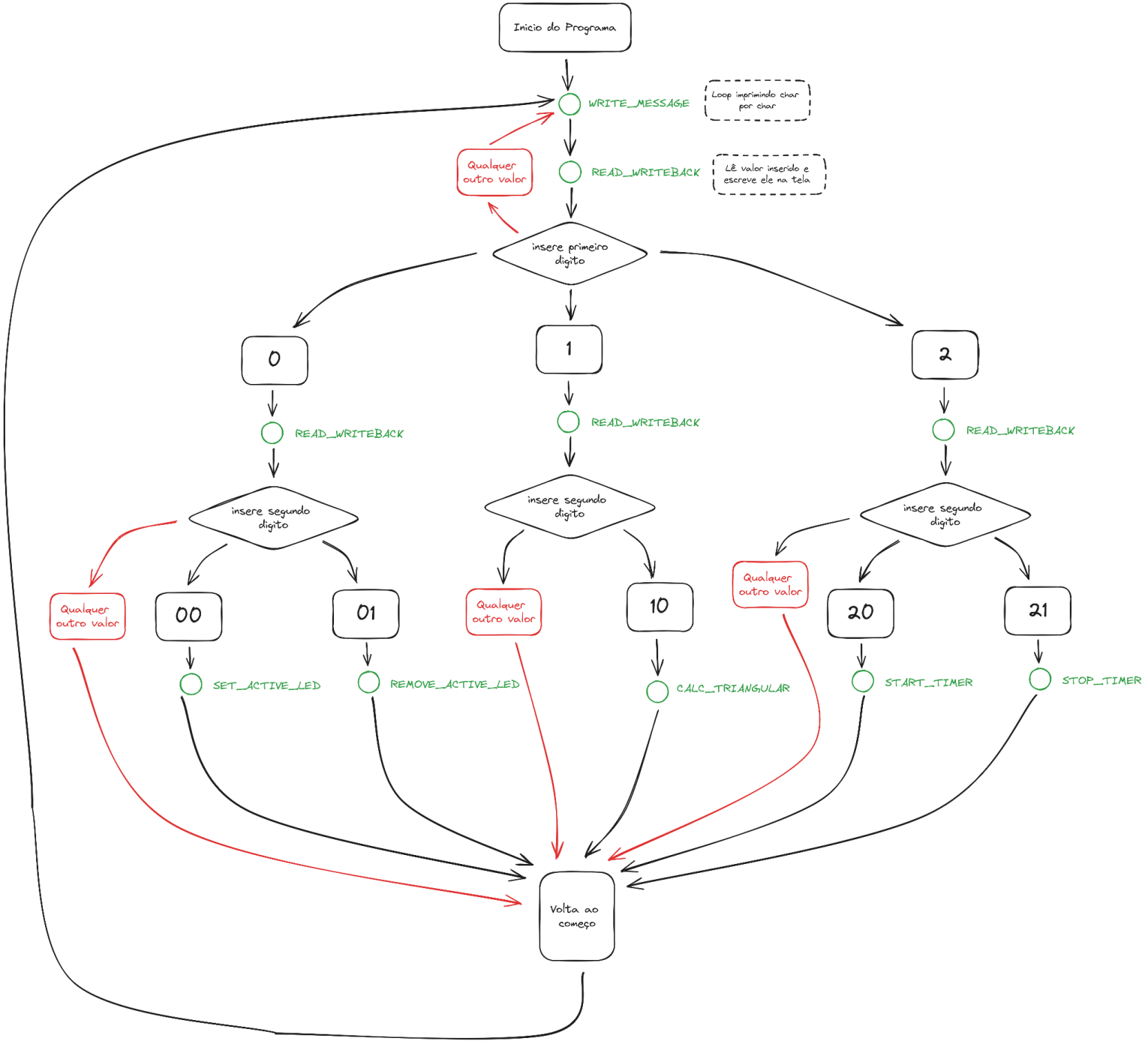
Para controlar o funcionamento das diferentes funções do sistema, foram utilizados quatro flags globais, nos registradores r4, r5, r6 e r7. As flags em r5 e r6 indicam o que deve ocorrer durante a interrupção do temporizador, sendo úteis para saber quando o temporizador deve ser iniciado e cancelado (visto que, se o cronômetro está ativo, não deve-se re-configurar o temporizador para efetuar a piscagem dos LEDs, por exemplo)

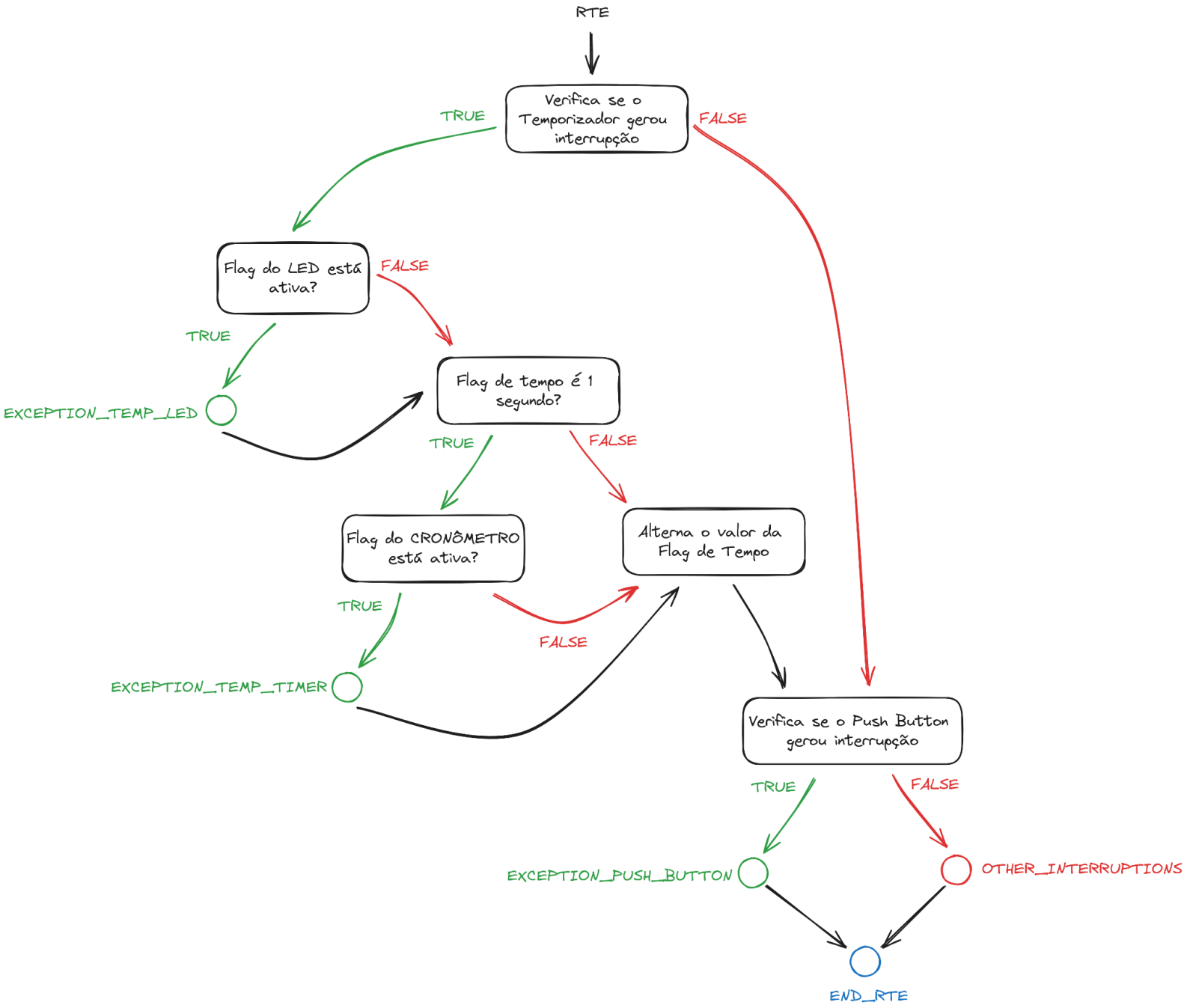
- r4: indica se a próxima interrupção é de 0,5s ou 1s. O temporizador é configurado para gerar interrupções a cada 0,5s, portanto a flag é importante para saber se a função do cronômetro deve ser chamada (caso ele esteja ativo), uma vez que o cronômetro deve ter seu valor alterado a cada 1s.
- r5: indica se há algum LED piscando. Quando essa flag vale 1, entende-se que na interrupção do temporizador, deve ser chamada a função para efetuar a piscagem dos LEDs.
- r6: indica se o cronômetro está funcionando. Nesse caso, a interrupção do temporizador chama a função para alterar o valor exibido no display de sete segmentos a cada um segundo.
- r7: indica se o cronômetro está pausado. Difere-se de r6, pois no caso do cronômetro pausado, não pode-se realizar operações de "limpeza" no temporizador ou no contador usado para armazenar o valor cronometrado.

4. Arquivos e Fluxogramas

4.1. main.s

Nesse arquivo temos o fluxo inicial do sistema, onde chamamos outras funções para a execução mediante a entrada do usuário. Na página seguinte é possível observar o fluxograma com o comportamento do sistema. Além disso temos a RTE (Rotina de tratamento de exceções) presente nesse arquivo invocando as outras funções para tratamento de rotinas específicas como, por exemplo o push button, e também é possível observar abaixo um fluxograma com o tratamento utilizado.


Inicio do Programa
WRITE_MESSAGE
Loop imprimindo char por char
READ_WRITEBACK
Lê valor inserido e escreve ele na tela
Qualquer outro valor
insere primeiro digito
0
1
2
READ_WRITEBACK
READ_WRITEBACK
READ_WRITEBACK
insere segundo digito
insere segundo digito
insere segundo digito
Qualquer outro valor
00
01
Qualquer outro valor
10
Qualquer outro valor
20
21
SET_ACTIVE_LED
REMOVE_ACTIVE_LED
CALC_TRIANGULAR
START_TIMER
STOP_TIMER
Volta ao começo


RTE
Verifica se o Temporizador gerou interrupção
TRUE
FALSE
Flag do LED está ativa?
FALSE
TRUE
EXCEPTION_TEMP_LED
Flag de tempo é 1 segundo?
TRUE
FALSE
Flag do CRONÔMETRO está ativa?
Alterna o valor da Flag de Tempo
TRUE
FALSE
EXCEPTION_TEMP_TIMER
Verifica se o Push Button gerou interrupção
TRUE
FALSE
EXCEPTION_PUSH_BUTTON
OTHER_INTERRUPTIONS
END_RTE

4.2. write_message.s

Esse arquivo contém a função de mesmo nome que se encarrega de exibir na UART a mensagem: *"Entre com o comando: "*.

4.3. read_writeback.s

A função *read_writeback* é responsável por ler o que o usuário digita na UART, exibindo de volta o que foi escrito. Faz uso de **polling** para saber quando é possível ler/escrever no dispositivo I/O.

4.4. set_active_led.s

Em *set_active_led*, a função *read_writeback* é chamada para obter o número do LED que deve ser aceso. Esse LED é então salvo no vetor *ACTIVE_LEDS* da memória (que armazena todos os LEDs que estão piscando). Se o vetor estiver vazio e o cronômetro não estiver ativo (uso da flag), o temporizador é configurado. Além disso, a flag referente aos LEDs é colocada em 1.

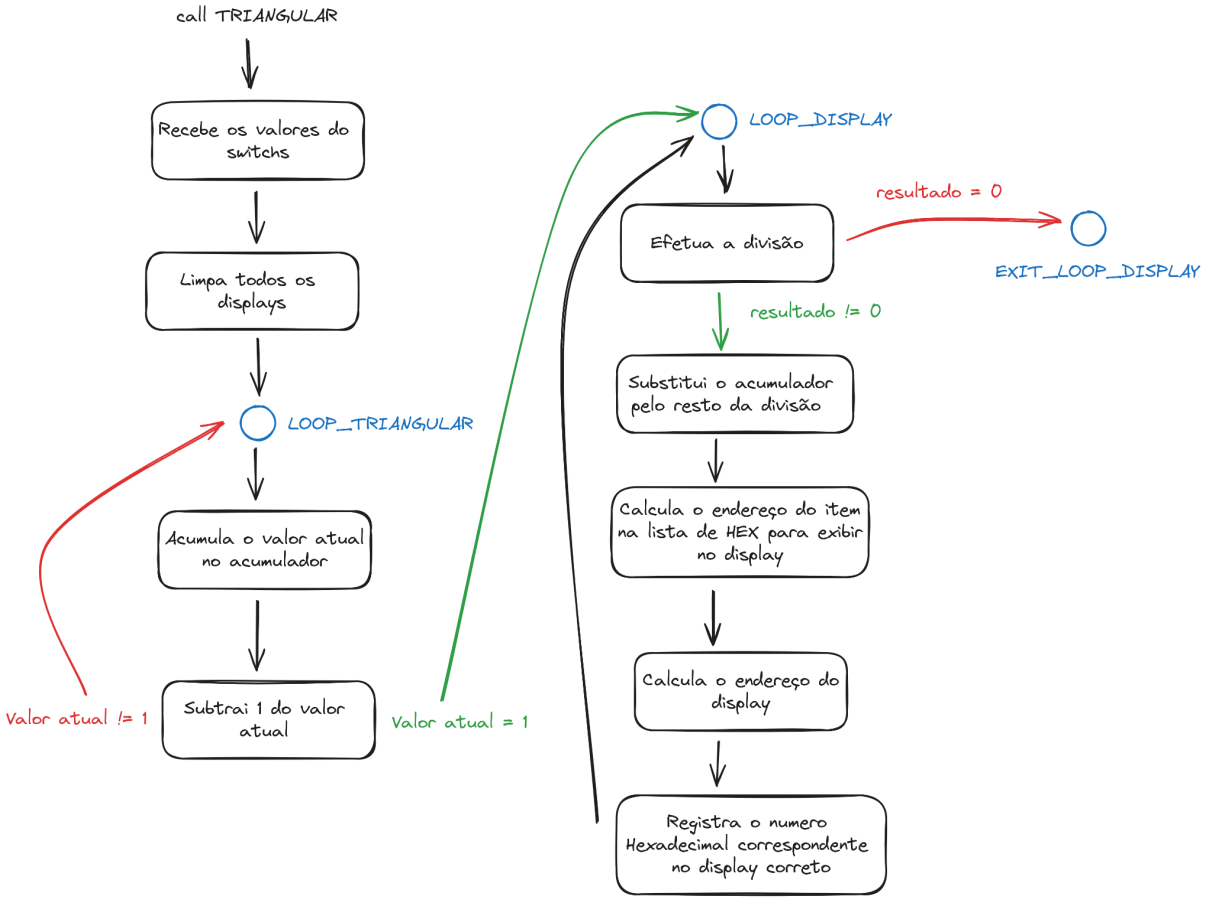
4.5. remove_active_led.s

O método *remove_active_led* chama *read_writeback*, para obter o número do LED, e o remove do vetor *ACTIVE_LEDS.* Se o vetor ficar vazio e o cronômetro não estiver ativo, limpa o temporizador, apagando todos os LEDs e colocando em 0 a flag referente aos LEDs.

4.6. triangular.s

Neste método chamado de triangular temos o responsável por fazer a lógica do comando ‘10’ de nosso projeto. Durante o seu processamento ele obtém inicialmente o valor dos switches ativados o qual será calculado posteriormente, depois disso também zera e apaga todos os displays de maneira a garantir que o número que será mostrado na chamada dessa função não seja mostrado incorretamente devido a algum outro valor que esteja no display. Em seguida, ele inicia um simples loop de maneira que seja calculado de fato o valor triangular do número lido nos switches e, ao obter esse valor e sair desse primeiro loop, também dá-se início a um segundo, o qual obtém dígito a dígito do número triangular obtido e por fim mostra no display de 7 segmentos esse algarismo, sendo cada umas das iterações desse segundo loop responsável por pegar um dígito e acender um respectivo display. É importante ressaltar também que a lógica utilizada para obter cada algarismo foi a seguinte:

A partir do número inicial a ser mostrado no display são feitas inúmeras divisões por 10, onde o resto da divisão compreende como o respectivo algarismo que deve ser mostrado de 0 a 9 no display de 7 segmentos. Além disso o quociente de cada divisão se torna o dividendo da próxima iteração, tal loop se encerra quando o quociente chega por fim ao valor 0.



4.7. start_timer.s

Esse método é responsável por iniciar o cronômetro. Nele, a flag referente ao cronômetro é colocada em 1 e o contador do temporizador é limpado (ficando com o valor zero). Se não houver nenhum LED piscando, é configurado o temporizador.

4.8. stop_timer.s

Chamada para encerrar o cronômetro, a função limpa a flag do cronômetro, o display de sete segmentos e o temporizador (caso nenhum LED esteja piscando).

4.9. start_temp_counter.s

Método que configura o temporizador, definindo que a as interrupções ocorrerão a cada 500ms. Configura também a flag r4, indicando que a próxima interrupção será de 0,5s.

4.10. exception_temp_led.s

Chamada na RTE, caso a interrupção seja do temporizador e a flag r5 esteja ativa. Nessa função, o vetor *ACTIVE_LEDS* é percorrido, e para cada LED que está no vetor, seu valor é obtido (a partir do endereço base dos LEDs vermelhos) e invertido, de modo que os LEDs escolhidos piscam.

4.11. exception_temp_timer.s

Chamada na RTE, caso a interrupção seja do temporizador, a flag r4 esteja em 1 (ou seja, a interrupção é de 1s) e a flag r6 também (indicando que o cronômetro está ativo). A função retorna caso a flag r7 indique que a contagem está pausada. Caso contrário, o acumulador do cronômetro é incrementado na memória e o novo valor é mostrado no display de sete segmentos.

4.12. exception_push_button.s

Chamada na RTE se a interrupção foi causada pelo *push button*. Apenas inverte a flag r7, indicando se o cronômetro está ou não pausado.

5. Conclusão

Como conclusão de nosso relatório podemos citar o quão proveitoso o projeto foi, de maneira que através do desenvolvimento conseguimos perceber que utilizamos muitos senão todos os pontos vistos durante os laboratórios e as aulas, de forma que foi possível sintetizar e aplicar todo o conteúdo visto da melhor maneira através dos comandos que foram implementados.

Por fim, também achamos importante destacar as dificuldades encontradas durante o desenvolvimento do projeto, sendo algumas delas por exemplo a conversão do número triangular para a visualização correta no display de 7 segmentos, onde tivemos problemas para encontrar uma maneira de demonstrar algarismo a algarismo do número,e que com a ajuda do professor e procurando mais sobre possíveis algoritmos de conversão como o Binary coded decimal, chegamos na solução explicada no tópico 4.6 de nosso relatório.

Outra dificuldade interessante de ser citada que enfrentamos foi o tratamento do comando de entrada do terminal que foi um dos primeiros impasses que enfrentamos, e que seria responsável por fazer a organização inicial do nosso código assim como as respectivas chamadas de funções, acabamos decidindo tomar um caminho mais sequencial e verificando número a número inserido no comando, levando a podemos dizer, uma certa árvore de escolhas que leva até a chamada de função de um dos comandos.

Como última dificuldade a ser citada temos o uso das 4 flags que citamos anteriormente, onde tivemos problemas para sincronizar as funções com essas flags, de maneira que nenhuma atrapalhasse o funcionamento da outra, e também que o cronômetro e o piscar dos leds atuasse de maneira conjunta principalmente por termos optado como estratégia as flags sendo feitas em registradores específicos ao invés de em memória.