Relatório – Entrega Final Projeto 1 – Relógio

Alunos: Ana Carolina Souza e João Pedro Farias

A arquitetura do projeto é Registrador-Memoria, de acordo com as especificações do projeto. A conexão dos periféricos é parecida ao esquema mostrado na aula 8, porém com o processador funcionando com um banco de registradores no lugar do acumulador, de acordo com a necessidade da arquitetura.

O projeto conta com 16 instruções: as 11 vistas em aula, o ANDI que foi adicionado na entrega intermediária e 4 novas instruções: CGT (Compare Greater Than), JGT (Jump Greater Than), ADDI (soma com imediato) e SUBI (subtração com o imediato).

A sintaxe de CGT em assembly pede um endereço da memória na primeira posição (@4) e um registrador (R0) para efetuar a comparação. JGT requere algo parecido, com o endereço a ser saltado no código, porém a segunda posição do registrador é desnecessária.

Para ADDI e SUBI, a primeira posição pede o número que será somado ou subtraído ($1) e o registrador que contêm o valor a ser operacionalizado e o resultado armazenado (R1).

Os pontos de controle são parecidos com os vistos em aula, com a adição de dois novos pontos ao final: habFlagMaior (habilitar o flip flop para ser escrito com o resultado da comparação quando a instrução for CGT) e JGT (Para habilitar o mux seletor para fazer o salto quando a flag e a instrução de maior foram selecionadas). A sua distribuição, portanto, fica assim no decodificador de instruções:

alias habEscritaMEM : std\_logic is saida(0);

alias habLeituraMEM : std\_logic is saida(1);

alias habFlagIgual : std\_logic is saida(2);

alias operacao : std\_logic\_vector(1 downto 0) is saida(4 downto 3);

alias habA : std\_logic is saida(5);

alias SelMUX : std\_logic is saida(6);

alias JEQ\_c : std\_logic is saida(7);

alias JSR\_c : std\_logic is saida(8);

alias RET\_c : std\_logic is saida(9);

alias JMP\_c : std\_logic is saida(10);

alias habEscritaRetorno : std\_logic is saida(11);

alias habFlagMaior : std\_logic is saida(12);

alias JGT\_c : std\_logic is saida(13);

O formato das instruções vistas em aula fica igual, uma vez não alterada as posições dos pontos de controle. Portanto apenas as instruções novas terão os seus pontos de controle demonstrados:

Table

Description automatically generated

Para fazer o relógio funcionar, uma mudança no processamento do clock foi necessário. Dois divisores de clock foram adicionados na saída do KEY0, conforme a foto abaixo:

Diagram, schematic

Description automatically generated

Esses divisores servem para simular o pressionamento do KEY0 como se fosse uma vez por segundo, no caso do divisorN ou a um ritmo mais acelerado para o divisorA. O divisorN usa 25.000.000Hz na sua divisão e o divisorA usa 100.000Hz. Ambas as entradas entram num mux para selecionar qual será o simulador de KEY0, que é o muxClock da figura. O seletor desse mux é o próprio KEY0. Portanto, se o KEY0 não está apertado, o display irá incrementar uma vez por segundo. Mas, se o KEY0 está pressionado, o incremento será ligeiramente mais rápido, requisito do projeto para a rápida verificação. Um ponto importante para destacar é que para o KEY0 funcionar quando ele é segurado, foi preciso tirar o edge detector presente da última entrega. Um diagrama com todas as conexões do projeto estará em anexo na pasta de entrega desse projeto com o nome “Conexões\_Projeto.pdf”

Anexo – Mapa de Memória

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Endereço em Decimal** | **Periférico** | **Largura dos Dados** | **Tipo de Acesso** | **Bloco (Página) de Memória** |
| 0 ~ 63 | RAM | 8 bits | Leitura/Escrita | 0 |
| 64 ~ 127 | Reservado | – | – | 1 |
| 128 ~ 191 | Reservado | – | – | 2 |
| 192 ~ 255 | Reservado | – | – | 3 |
| 256 | LEDR0 ~ LEDR7 | 8 bits | Escrita | 4 |
| 257 | LEDR8 | 1 bit | Escrita | 4 |
| 258 | LEDR9 | 1 bit | Escrita | 4 |
| 259 ~ 287 | Reservado | – | – | 4 |
| 288 | HEX0 | 4 bits | Escrita | 4 |
| 289 | HEX1 | 4 bits | Escrita | 4 |
| 290 | HEX2 | 4 bits | Escrita | 4 |
| 291 | HEX3 | 4 bits | Escrita | 4 |
| 292 | HEX4 | 4 bits | Escrita | 4 |
| 293 | HEX5 | 4 bits | Escrita | 4 |
| 294 ~ 319 | Reservado | – | – | 4 |
| 320 | SW0 ~ SW7 | 8 bits | Leitura | 5 |
| 321 | SW8 | 1 bit | Leitura | 5 |
| 322 | SW9 | 1 bit | Leitura | 5 |
| 323 ~ 351 | Reservado | – | – | 5 |
| 352 | KEY0 | 1 bit | Leitura | 5 |
| 353 | KEY1 | 1 bit | Leitura | 5 |
| 354 | KEY2 | 1 bit | Leitura | 5 |
| 355 | KEY3 | 1 bit | Leitura | 5 |
| 356 | FPGA\_RESET | 1 bit | Leitura | 5 |
| 357 ~ 383 | Reservado | – | – | 5 |
| 384 ~ 447 | Reservado | – | – | 6 |
| 448 ~ 510 | Reservado | – | – | 7 |
| 510 | Limpa Leitura KEY1 | – | Escrita | 7 |
| 511 | Limpa Leitura KEY0 | – | Escrita | 7 |

Anexo – Mapa da Memória RAM

|  |  |  |
| --- | --- | --- |
| **Endereço em Decimal** | **Variável** | **Significado** |
| 0 | SUNI | Unidade dos Segundos |
| 1 | SDEC | Decimal dos Segundos |
| 2 | MUNI | Unidade dos Minutos |
| 3 | MDEC | Decimal dos Minutos |
| 4 | HUNI | Unidade da Hora |
| 5 | HDEC | Decimal da Hora |
| 7 | VAR1 | Guarda 1 |
| 8 | VAR10 | Guarda 10 |
| 9 | VAR6 | Guarda 6 |
| 10 | VAR3 | Guarda 3 |
| 11 | VAR5 | Guarda 5 |
| 12 | VAR9 | Guarda 9 |
| 13 | VAR4 | Guarda 4 |
| 14 | VAR2 | Guarda 2 |

Anexo – Manual de Instruções

Periféricos usados

- KEY0: Aumentar a frequência

- KEY1: Configurar o horário

- SW0-3: Switches para configuração do horário

- LEDR0-7: Leds para indicar casas dos minutos e horas ao configurar o horário

Aumentar a frequência:

- Segurar KEY0 até chegar até o horário desejado

Configurar o horário:

- Apertar KEY1 para iniciar

- Display HEX congela

- Usar SW3 até SW0 para escolher o limite para a casa decimal indicada pelo led

- Display HEX atualiza com o número novo

- Apertar KEY1 para avançar para a próxima casa decimal

- Repetir as últimas duas etapas até configurar o decimal da hora e o relógio voltará a contar, desligando o LEDR

Anexo – Código usado em Assembly (Assembler incluído em anexo na pasta)

Setup :

LDI $0, R7

STA @HEX0, R7 #Limpando os 7 segmentos

STA @HEX1, R7

STA @HEX2, R7

STA @HEX3, R7

STA @HEX4, R7

STA @HEX5, R7

STA @LEDR7, R7 #Limpando os leds

STA @LEDR8, R7

STA @LEDR9, R7

STA @SUNI, R7 #Limpando os 7 segmentos

STA @SDEC, R7

STA @MUNI, R7

STA @MDEC, R7

STA @HUNI, R7

STA @HDEC, R7

STA @CLR0, R7 #Limpar KEY0

STA @CLR1, R7 #Limpar KEY1

STA @CLR2, R7 #Limpar KEY2

STA @CLR3, R7 #Limpar KEY3

STA @CLRR, R7 #Limpar FPGA\_RESET

LDI $1, R0

STA @VAR1, R0 #Inicializando constante com 1

LDI $10, R0

STA @VAR10, R0 #Inicializando constante com 10

LDI $6, R0

STA @VAR6, R0 #Inicializando constante com 6

LDI $3, R0

STA @VAR3, R0 #Inicializando constante com 2

LDI $5, R0

STA @VAR5, R0 #Inicializando constante com 5

LDI $9, R0

STA @VAR9, R0 #Inicializando constante com 9

LDI $4, R0

STA @VAR4, R0 #Inicializando constante com 4

LDI $2, R0

STA @VAR2, R0 #Inicializando constante com 2

Loop :

LDA @KEY0, R0 #Ler KEY0

ANDI @1, R0 #Aplicar máscara a leitura da KEY

CEQ @VAR1, R0 #Comparar com 1

JEQ @HubIncremento #Pular pra subrotina de incremento

PosIncremento :

LDA @KEY1, R0 #Ler KEY1

ANDI @1, R0 #Aplicar máscara a leitura da KEY

CEQ @VAR1, R0 #Comparar com 1

JEQ @HubLimite #Pular pra subrotina de limite

PosLimite :

JSR @Verificar #Pular pra subrotina de verificar limite

LDA @SUNI, R0 #Carrega acumulador com valor da unidade dos segundos

STA @HEX0, R0 #Carrega unidade no HEX0

LDA @SDEC, R0 #Carrega acumulador com valor da dezena dos segundos

STA @HEX1, R0 #Carrega unidade no HEX1

LDA @MUNI, R0 #Carrega acumulador com valor da unidade dos minutos

STA @HEX2, R0 #Carrega unidade no HEX2

LDA @MDEC, R0 #Carrega acumulador com valor da dezena dos minutos

STA @HEX3, R0 #Carrega unidade no HEX3

LDA @HUNI, R0 #Carrega acumulador com valor da unidade da hora

STA @HEX4, R0 #Carrega unidade no HEX4

LDA @HDEC, R0 #Carrega acumulador com valor da dezena da hora

STA @HEX5, R0 #Carrega unidade no HEX5

JMP @Loop #Reiniciar Loop

HubIncremento :

JSR @Incremento #Para usar RET

JMP @PosIncremento #Voltar pro loop

HubLimite :

JSR @Limite #Para usar RET

JMP @PosLimite #Voltar pro loop

Incremento :

STA @CLR0, R7

LDA @SUNI, R0 #Carrega R0 com unidade dos segundos

ADDI $1, R0 #Soma 1 com R0

CEQ @VAR10, R0 #Compara com 10

JEQ @SDezena #Caso 10, pular pra dezena dos segundos

STA @SUNI, R0 #Caso contrario, salvar na unidade dos segundos

RET #Retornar

SDezena :

STA @SUNI, R7 #Zera unidade dos segundos

LDA @SDEC, R0 #Carrega dezena dos segundos no R0

ADDI $1, R0 #Soma 1 com R0

CEQ @VAR6, R0 #Compara com 6

JEQ @MUnidade #Caso 6, pular pra unidade dos minutos

STA @SDEC, R0 #Caso contrario, salvar na dezena dos minutos

RET #Retornar

MUnidade :

STA @SDEC, R7 #Zera dezena dos segundos

LDA @MUNI, R0 #Carrega unidade dos minutos em R0

ADDI $1, R0 #Soma 1 com R0

CEQ @VAR10, R0 #Compara com 10

JEQ @MDezena #Caso 10, pular pra dezena dos minutos

STA @MUNI, R0 #Caso contrario, salvar na centena

RET #Retornar

MDezena :

STA @MUNI, R7 #Zera unidade dos minutos

LDA @MDEC, R0 #Carrega dezena dos minutos em R0

ADDI $1, R0 #Soma 1 com R0

CEQ @VAR6, R0 #Compara com 6

JEQ @HUnidade #Caso 6, pular pra unidade da hora

STA @MDEC, R0 #Caso contrario, salvar na dezena dos minutos

RET #Retornar

HUnidade :

STA @MDEC, R7 #Zera dezena dos minutos

LDA @HUNI, R0 #Carrega unidade da hora em R0

ADDI $1, R0 #Soma 1 com R0

CEQ @VAR10, R0 #Compara com 10

JEQ @HDezena #Caso 10, pular pra dezena da hora

STA @HUNI, R0 #Caso contrario, salvar na unidade da hora

RET #Retornar

HDezena :

STA @HUNI, R7 #Zera unidade da hora

LDA @HDEC, R0 #Carrega dezena da hora em R0

ADDI $1, R0 #Soma 1 com R0

RET #Retornar

Limite :

STA @CLR1, R7 #Limpar KEY1

LDI $4, R0

STA @LEDR7, R0 #Liga LEDR2 pra indicar aguardando leitura

LimiteMUni :

STA @CLR1, R7 #Limpar KEY1

LDA @SW7, R1 #Leitura do SW0-7

ANDI @15, R1 #Aplicar mascara a leitura do botao para só pegar SW(3 downto 0)

CGT @VAR10, R1 #Verificar overflow do HEX

JGT @LimiteMUniMax

PosLimiteMUni :

STA @MUNI, R1 #Guarda na memoria valor da unidade dos minutos

STA @HEX2, R1 #Carrega leitura do SW0-7 no HEX2

LDA @KEY1, R0 #Ler KEY1

ANDI @1, R0 #Aplicar máscara a leitura da KEY

CEQ @VAR1, R0 #Ve se KEY1 está apertado

STA @CLR1, R7 #Limpar KEY1

JEQ @PosLMU #Se tiver, guarda e vai embora

JMP @LimiteMUni #Caso contrario, faz de novo

PosLMU :

LDI $8, R0

STA @LEDR7, R0 #Liga LEDR3 pra indicar aguardando leitura

LimiteMDec :

STA @CLR1, R7 #Limpar KEY1

LDA @SW7, R1 #Leitura do SW0-7

ANDI @7, R1 #Aplicar mascara a leitura do botao para só pegar SW(2 downto 0)

CGT @VAR6, R1 #Verificar overflow do HEX

JGT @LimiteMDecMax

PosLimiteMDec :

STA @MDEC, R1 #Guarda na memoria valor da dezena dos minutos

STA @HEX3, R1 #Carrega leitura do SW0-7 no HEX3

LDA @KEY1, R0 #Ler KEY1

ANDI @1, R0 #Aplicar máscara a leitura da KEY

CEQ @VAR1, R0 #Ve se KEY1 está apertado

STA @CLR1, R7 #Limpar KEY1

JEQ @PosLMD #Se tiver, guarda e vai embora

JMP @LimiteMDec #Caso contrario, faz de novo

PosLMD :

LDI $16, R0

STA @LEDR7, R0 #Liga LEDR4 pra indicar aguardando leitura

LimiteHUni :

STA @CLR1, R7 #Limpar KEY1

LDA @SW7, R1 #Leitura do SW0-7

ANDI @15, R1 #Aplicar mascara a leitura do botao para só pegar SW(3 downto 0)

CGT @VAR10, R1 #Verificar overflow do HEX

JGT @LimiteHUniMax

PosLimiteHUni :

STA @HUNI, R1 #Guarda na memoria valor da unidade da hora

STA @HEX4, R1 #Carrega leitura do SW0-7 no HEX4

LDA @KEY1, R0 #Ler KEY1

ANDI @1, R0 #Aplicar máscara a leitura da KEY

CEQ @VAR1, R0 #Ve se KEY1 está apertado

STA @CLR1, R7 #Limpar KEY1

JEQ @PosLHU #Se tiver, guarda e vai embora

JMP @LimiteHUni #Caso contrario, faz de novo

PosLHU :

LDI $32, R0

STA @LEDR7, R0 #Liga LEDR5 pra indicar aguardando leitura

LDA @HUNI, R0 #Verificar unidade da hora -> se for maior que 4, nao pode colocar 2 na hora

CGT @VAR4, R0 #Verificar overflow do HEX

JGT @LimiteHDecRestringe

LimiteHDec :

STA @CLR1, R7 #Limpar KEY1

LDA @SW7, R1 #Leitura do SW0-7

ANDI @3, R1 #Aplicar mascara a leitura do botao para só pegar SW(1 downto 0)

CEQ @VAR3, R1 #Verificar overflow do HEX

JEQ @LimiteHDecMax

PosLimiteHDec :

STA @HDEC, R1 #Guarda na memoria valor da unidade da hora

STA @HEX5, R1 #Carrega leitura do SW0-7 no HEX4

LDA @KEY1, R0 #Ler KEY1

ANDI @1, R0 #Aplicar máscara a leitura da KEY

CEQ @VAR1, R0 #Ve se KEY1 está apertado

STA @CLR1, R7 #Limpar KEY1

JEQ @PosLHD #Se tiver, guarda e vai embora

JMP @LimiteHDec #Caso contrario, faz de novo

PosLHD :

STA @LEDR7, R7 #Desliga LEDR0-7

RET

LimiteHDecRestringe :

STA @CLR1, R7 #Limpar KEY1

LDA @SW7, R1 #Leitura do SW0-7

ANDI @1, R1 #Aplicar mascara a leitura do botao para só pegar SW(3 downto 0)

STA @HDEC, R1 #Guarda na memoria valor da unidade da hora

STA @HEX5, R1 #Carrega leitura do SW0-7 no HEX4

LDA @KEY1, R0 #Ler KEY1

ANDI @1, R0 #Aplicar máscara a leitura da KEY

CEQ @VAR1, R0 #Ve se KEY1 está apertado

STA @CLR1, R7 #Limpar KEY1

JEQ @PosLHDR #Se tiver, guarda e vai embora

JMP @LimiteHDecRestringe #Caso contrario, faz de novo

PosLHDR :

STA @LEDR7, R7 #Desliga LEDR0-7

RET

LimiteMUniMax :

LDI $9, R1 #Carrega 9 no R1

JMP @PosLimiteMUni #Volta

LimiteMDecMax :

LDI $5, R1 #Carrega 5 no R1

JMP @PosLimiteMDec #Volta

LimiteHUniMax :

LDI $9, R1 #Carrega 9 no R1

JMP @PosLimiteHUni #Volta

LimiteHDecMax :

LDI $2, R1 #Carrega 2 no R1

JMP @PosLimiteHDec #Volta

Verificar :

LDA @SUNI, R3 #Carrega valor da unidade dos segundos

CEQ @VAR9, R3 #Compara unidade com o limite

JEQ @VerificarSDec #Se estiver no limite, checar dezena

RET

VerificarSDec :

LDA @SDEC, R3 #Carrega valor da dezena

CEQ @VAR5, R3 #Compara dezena com o limite

JEQ @VerificarMUni #Se estiver no limite, checar centena

RET

VerificarMUni :

LDA @MUNI, R3 #Carrega valor da centena

CEQ @VAR9, R3 #Compara centena com o limite

JEQ @VerificarMDec #Se estiver no limite, checar unidade de milhar

RET

VerificarMDec :

LDA @MDEC, R3 #Carrega valor da unidade de milhar

CEQ @VAR5, R3 #Compara unidade de milhar com o limite

JEQ @VerificarHUni #Se estiver no limite, checar dezena de milhar

RET

VerificarHUni :

LDA @HUNI, R3 #Carrega valor da dezena de milhar

CEQ @VAR3, R3 #Compara dezena de milhar com o limite

JEQ @VerificarHDec #Se estiver no limite, checar centena de milhar

RET

VerificarHDec :

LDA @HDEC, R3 #Carrega valor da centena de milhar

CEQ @VAR2, R3 #Compara centena de milhar com o limite

JEQ @LimiteAtingido #Se estiver no limite, pular pra SR de limite atingido

RET

LimiteAtingido :

STA @SUNI, R7 #Limpando os 7 segmentos

STA @SDEC, R7

STA @MUNI, R7

STA @MDEC, R7

STA @HUNI, R7

STA @HDEC, R7

Retorno :

RET