Design de Computadores

P1.2 – Entrega Final - Relógio

Henrique Martinelli Frezzatti

Lívia Sayuri Makuta

Nicolas Maciel Queiroga

São Paulo, 1 de novembro de 2022

1) Introdução

Esse relatório tem como objetivo explicar alguns detalhes e informações acerca do projeto final envolvendo a entrega de um relógio funcional a partir da entrega anterior intermediária do contador.

O documento será dividido nos seguintes tópicos: modo de uso, arquitetura do relógio e seu funcionamento, formato das instruções, os pontos de controle e o mapa de memória atualizado.

2) Modo de uso

O modo de uso do relógio implementado pode ser visto no vídeo a seguir: link. Além disso, todos os passos também estão descritos abaixo:

Inicialmente, quando a FPGA é carregada com o código, por predefinição, o relógio começará a contar normalmente, estando configurado no modo 24h, contando, portanto, de 0h até 23h59min. Ao clicar em KEY0, o relógio irá começar a acelerar a contagem – de tal forma que 1 segundo é como se fossem 2000 segundos. Para resetar a contagem do relógio, podemos pressionar o botão FPGA_RESET e, assim, reiniciando a contagem a partir de 0 horas.

O relógio projetado nesse trabalho também possui a funcionalidade de despertador e, com isso, é possível selecionar um horário qualquer de acordo com a vontade do usuário e, para isso, pressiona-se o botão KEY1. Após pressioná-lo pela primeira vez, o LED0 irá acender, indicando que está no modo de configuração da unidade de segundo e, com isso, podemos selecionar as chaves necessárias em binário (SW0 a SW3) para determinar o número para a unidade dos segundos. Feito isso, pressionamos o botão KEY1 novamente, acendendo agora o LED1, indicando que está no modo de configuração de despertador para a dezena dos segundos e, novamente, o mesmo processo é repetido até o usuário terminar de configurar todas as casas do despertador: segundos, minutos e horas. No vídeo, é configurado para o relógio despertar quando chegar em 0 horas, 0 minutos e 30 segundos.

Após a configuração do despertador, o relógio voltará à contagem normal, e quando atingir o horário configurado – no caso do vídeo de 00:00:30, o LED8 irá acender, indicando que despertou e o relógio continuará indicando o horário normalmente. Para desligar e desabilitar o despertador, basta pressionar o botão KEY2.

Além disso, também é possível configurar um novo horário inicial no relógio ao pressionar o botão KEY3. No entanto, assim como para configurar o horário do despertador, ele aceita apenas horas válidas, sendo o valor limite 24h. No vídeo, é testado o caso em que se configura tudo com o valor de número 4, o que resulta em um erro fazendo o LED9 acender. Nesse caso, basta observar o LED (do LED0 até o LED5) que foi aceso, o que indica que a partir dali a correção precisa ser feita. Por exemplo: se o usuário configurar o despertador ou o novo horário que sejam maiores que 24h, ele precisará configurar novamente os segundos, os minutos e as horas, mas, se ele configurar 24h com minutos que não estejam em 0, a configuração irá voltar para ele reescrever a partir dos minutos o novo horário.

3) Arquitetura do relógio

Para esta entrega final do relógio, fizemos o uso da arquitetura registrador-memória. Essa arquitetura já havia sido implementada na entrega anterior do contador, já visando a facilitação da implementação do relógio no trabalho atual.

4) Total de instruções e sua sintaxe

O total de instruções existentes no relógio são 16, e todas com sua sintaxe e funcionalidade se encontram na tabela a seguir:

TABELA MNEMÔNICOS		
MNEMÔNICO	**CÓDIGO BINÁRIO**	**DESCRIÇÃO**
NOP	0000	Não faz nada.
LDA	0001	Carrega um valor da memória para o acumulador.
SOMA	0010	Soma A e B e armazena em A (acumulador).
SUB	0011	Subtrai B de A e armazena em A (acumulador).
LDI	0100	Carrega o valor imediato no acumulador.
STA	0101	Salva o que está no acumulador para a memória.
JMP	0110	Desvio de execução incondicional.
JEQ	0111	Desvio de execução com condição de igual.
CEQ	1000	Compara se A é igual a B.
JSR	1001	Desvio para uma subrotina.
RET	1010	Retorno da subrotina.
ANDI	1011	Realiza operação lógica AND com o imediato.
CLT	1100	Compara se A é menor que B
JLT	1101	Desvio de execução se o valor do registrador for menor que o da memória
ADDI	1110	Realiza uma operação de SOMA com o registrador e o imediato
SUBI	1111	Realiza uma operação de SUBTRAÇÃO com o registrador e o imediato

Figura 1: Tabela de mnemônicos utilizados no contador, suas funções e o código binário correspondente. Observação: em relação a entrega anterior, 4 instruções foram adicionadas: o CLT, o JLT, ADDI e SUBI.

Nessa tabela estão as possíveis instruções a serem passadas para o processador com uma breve descrição e o seu mnemônico, que é como será acionada a instrução e a maneira com a qual são implementadas as funções em *assembly*, além de que também são usadas para facilitar a programação em VHDL.

5) Formato das instruções

As instruções apresentam o seguinte formato:

tmp(0) := instrução & RX & A8 & x”imediato”

No formato acima em VHDL, ‘instrução’ refere-se ao *opcode* da instrução, RX ao endereço do registrador que está sendo utilizado e A8 é o oitavo bit que é utilizado juntamente com o sétimo e o sexto bit para determinar com qual bloco de memória estamos trabalhando - essa parte ficará mais clara no tópico do diagrama do processador com os seus periféricos e no mapa de memória. Por fim, há também os 8 bits destinados ao valor imediato - onde o sexto e o sétimo bits citados anteriormente estão contidos, para decodificação do bloco correto.

A diferença em relação a entrega passada está nas novas instruções adicionadas: o CLT (“Compare if less than”), o JLT (“Jump if less than”), o ADDI e o SUBI.

Assim, por exemplo, na instrução:

tmp(0) := ADDI & R0 & ‘0’ & x"01";

Temos uma adição acontecendo entre o que está no registrador 0 com o valor que está no imediato. Se fosse um SUBI, ao invés da adição seria uma subtração.

Outro exemplo pode ser visto a seguir:

tmp(1) := CLT & R3 & '0' & x"13";
tmp(2) := JLT & R0 & ‘1’ & x”25”;

Na primeira instrução o valor que está carregado no registrador 3 é comparado para conferir se é menor que o valor que está na posição 19 da memória. E, na segunda instrução, se esse valor for menor, haverá um desvio para a linha 293 do código.

6) Diagrama do processador do relógio

Abaixo, pode ser observado o diagrama completo do processador construído e programado nesse projeto. Como pode ser observado, há novas conexões para duas novas instruções na palavra de controle (agora com 14 bits) e suas respectivas ligações no circuito, além da inserção de um novo Flip Flop para a nova Flag responsável pelo desvio JLT.

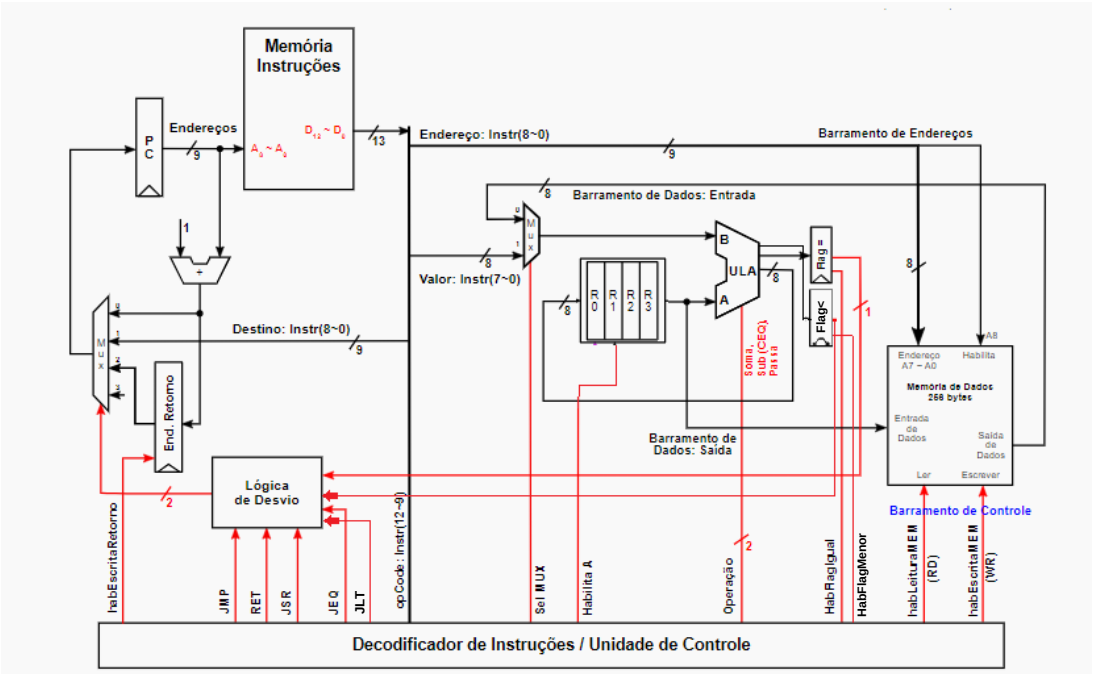

Figura 2: Diagrama do relógio montado com os novos pontos de controle.

7) Fluxo de dados para o processador

O fluxo de dados para processador, assim como o da entrega passada, pode ser representado da seguinte maneira:

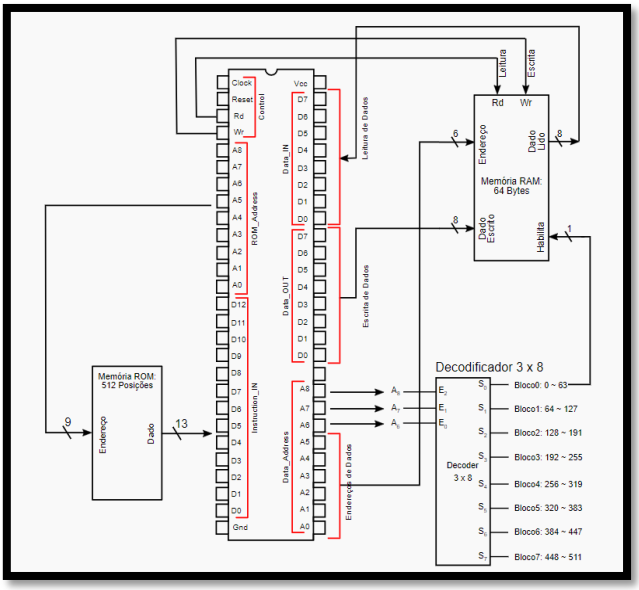

Figura 3 : Diagrama retirado do site da matéria feito pelo professor Paulo Carlos Santos. Nele temos a estrutura básica do contador com seus principais componentes: o processador, as memórias RAM e ROM e o decodificador 3x8.

No esquema acima, é possível observar que os endereços saem do processador por meio da variável ***ROM_Address*** - no caso, saem do *program counter* - e entram na memória ROM. Dessa forma, é o ***ROM_Address*** que informa à memória ROM qual será a próxima instrução. A partir da ROM, esses valores saem e entram novamente no processador por meio do sinal ***instruction_IN***, aonde depois vão para o MUX e, caso a memória esteja sendo utilizada, depois vão para a ULA. Da ULA, depois de realizar alguma operação (passa, soma, subtração e resultado de flags), o resultado vai ser guardado em algum dos quatro registradores: R0, R1, R2 e R3.

A partir do processador saem alguns sinais que se dirigem para memória e para os decodificadores de bloco de memória e de posição. Alguns desses são os sinais ***read*** e ***write*** que indicam se a memória RAM deve ser lida ou escrita - o que inclui botões ou chaves e LEDs ou displays de sete segmentos, respectivamente. Contudo, isso só acontecerá se o endereço dos LEDs ou display de sete segmentos estiverem na saída do ***Data_Address***, enquanto as chaves e botões apenas serão lidos se seus endereços estiverem no ***DATA_IN***.

8) Listagem dos pontos de controle e sua utilização

Para cada instrução há uma palavra de controle, a qual representa os bits que determinam os valores de cada ponto de controle do circuito para realizar a instrução específica. Abaixo se encontra as palavras de controle de cada instrução:

PALAVRA DE CONTROLE													
	HAB RET	JMP	RET	JSR	JEQ	JLT	SELMUX	HAB BLOCO	OP. ULA	FLAG =	FLAG <	READ	WRITE
NOP	0	0	0	0	0	0	X	0	XX	0	0	0	0
LDA	0	0	0	0	0	0	0	1	10	0	0	1	0
SOMA	0	0	0	0	0	0	0	1	01	0	0	1	0
SUB	0	0	0	0	0	0	0	1	00	0	0	1	0
ANDI	0	0	0	0	0	0	1	1	11	0	0	1	0
LDI	0	0	0	0	0	0	1	1	10	0	0	0	0
STA	0	0	0	0	0	0	0	0	XX	0	0	0	1
JMP	0	1	0	0	0	0	X	0	XX	0	0	0	0
JEQ	0	0	0	0	1	0	X	0	XX	0	0	0	0
CEQ	0	0	0	0	0	0	0	0	00	1	0	1	0
JSR	1	0	0	1	0	0	X	0	XX	0	0	0	0
RET	0	0	1	0	0	0	X	0	XX	0	0	0	0
CLT	0	0	0	0	0	0	0	0	00	0	1	1	0
JLT	0	0	0	0	0	1	X	0	XX	0	0	0	0
ADDI	0	0	0	0	0	0	1	1	01	0	0	0	0
SUBI	0	0	0	0	0	0	1	1	00	0	0	0	0

Figura 4: Tabela das palavras de controle de cada instrução utilizada no contador.

9) Diagrama de conexão do processador com os periféricos

Assim como na entrega passada, foram referenciados os mesmos periféricos na memória: LEDs, display de sete segmentos, chaves e botões. Vamos começar pelos periféricos de leitura. Antes, é importante salientar que estamos trabalhando com blocos de memória, isso porque desde a aula 8 nós dividimos as 512 posições de memória em 7 blocos (cada um com 64 posições), dessa forma, cada periférico está em algum desses blocos.

Portanto, para determinar o endereço de chaves e botões foram utilizados decodificadores 3x8 e portas AND, e isso foi necessário porque o bloco 5 é responsável por controlar ambos. Assim, para saber se é uma chave ou botão, além do decodificador que determina qual é o bloco de memória sendo acessado (nesse caso o 5), também temos outro decodificador que determina a posição dentro desse bloco. Mas para que tudo isso funcione também é preciso garantir que algumas condições estão sendo atendidas simultaneamente, e é por isso que a porta AND é utilizada. Abaixo está o diagrama que representa tudo o que foi descrito:

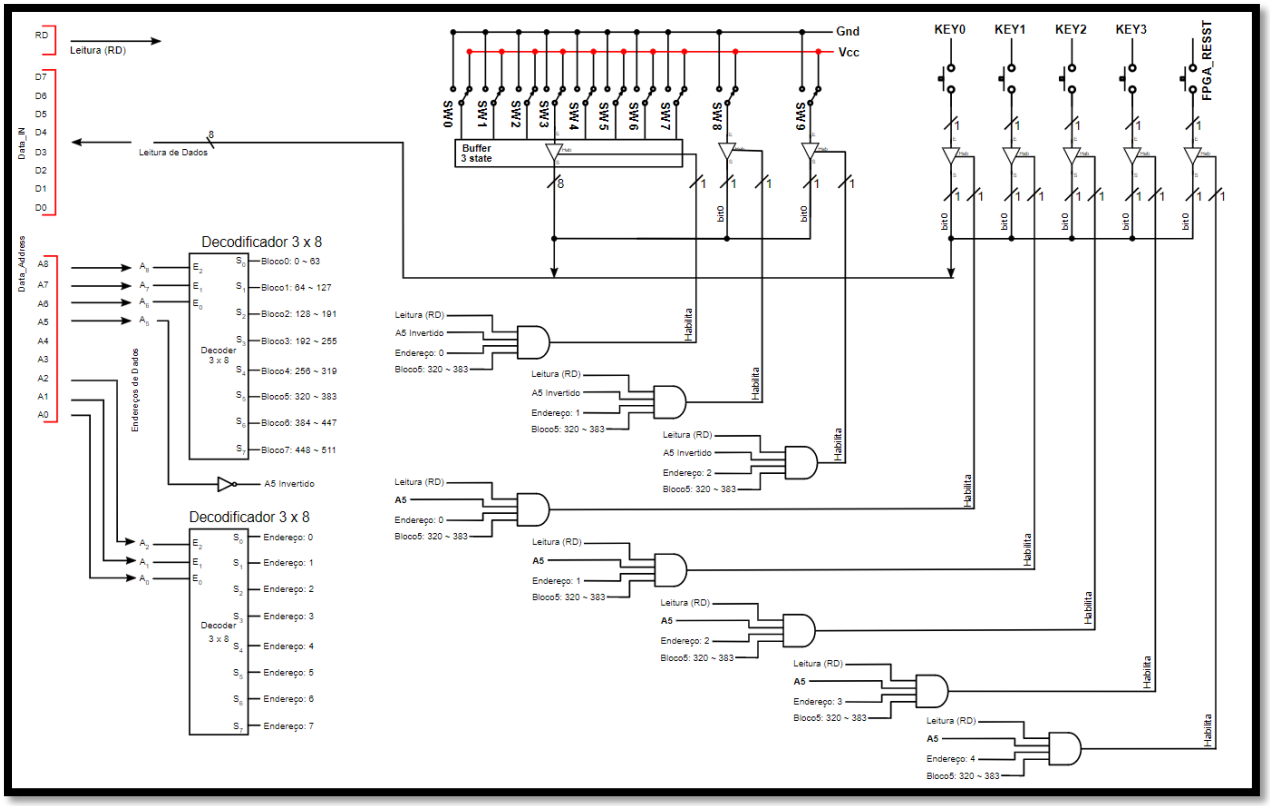

Figura 5: Diagrama retirado do site da matéria feito pelo professor Paulo Carlos Santos. Nele está a lógica de endereçamento de dois periféricos de leitura: as chaves e os botões.

Vale salientar que para a construção do relógio foi utilizada como referência o componente da base de tempo do site da matéria. Nesse componente há uma divisão do ciclo ativo de saída de 50%, sendo que para isso existe um contador que irá contar até a metade do valor do clock de 50 MHz e que depois irá ativar um flip flop que divide essa saída por 2. Dessa forma, foi possível chegar em 1s (frequência de 1 Hz). Para acelerar a contagem a mesma lógica foi implementada, porém com um divisor diferente. E para escolher entre a contagem mais acelerada ou a contagem normal, foi acrescentado um MUX cujo seletor é o próprio KEY0. Assim, se ele não estiver sendo pressionado, a contagem será normal, e se ele for pressionado, a contagem será acelerada.

Para os LEDs e o displays de 7 segmentos foi utilizado o mesmo raciocínio dos periféricos anteriores, de tal forma que os mesmos dois decodificadores 3x8 utilizados para o endereçamento de chaves e botões, também são utilizados para os LEDs e o display. Contudo, nesse caso, é utilizado o bloco 4 para o endereçamento em questão, utilizando diferentes endereços combinando com portas AND para diferenciar esses dois periféricos de escrita. Assim como salientado na entrega passada, para ativar os LEDs e os displays com os valores provenientes dos registradores, antes esses valores

precisam estar guardados em registradores ou flip-flops que apenas se ativados por meio desse endereçamento escreverão nos LEDs e display os valores indicados. Abaixo há a representação gráfica do que foi descrito:

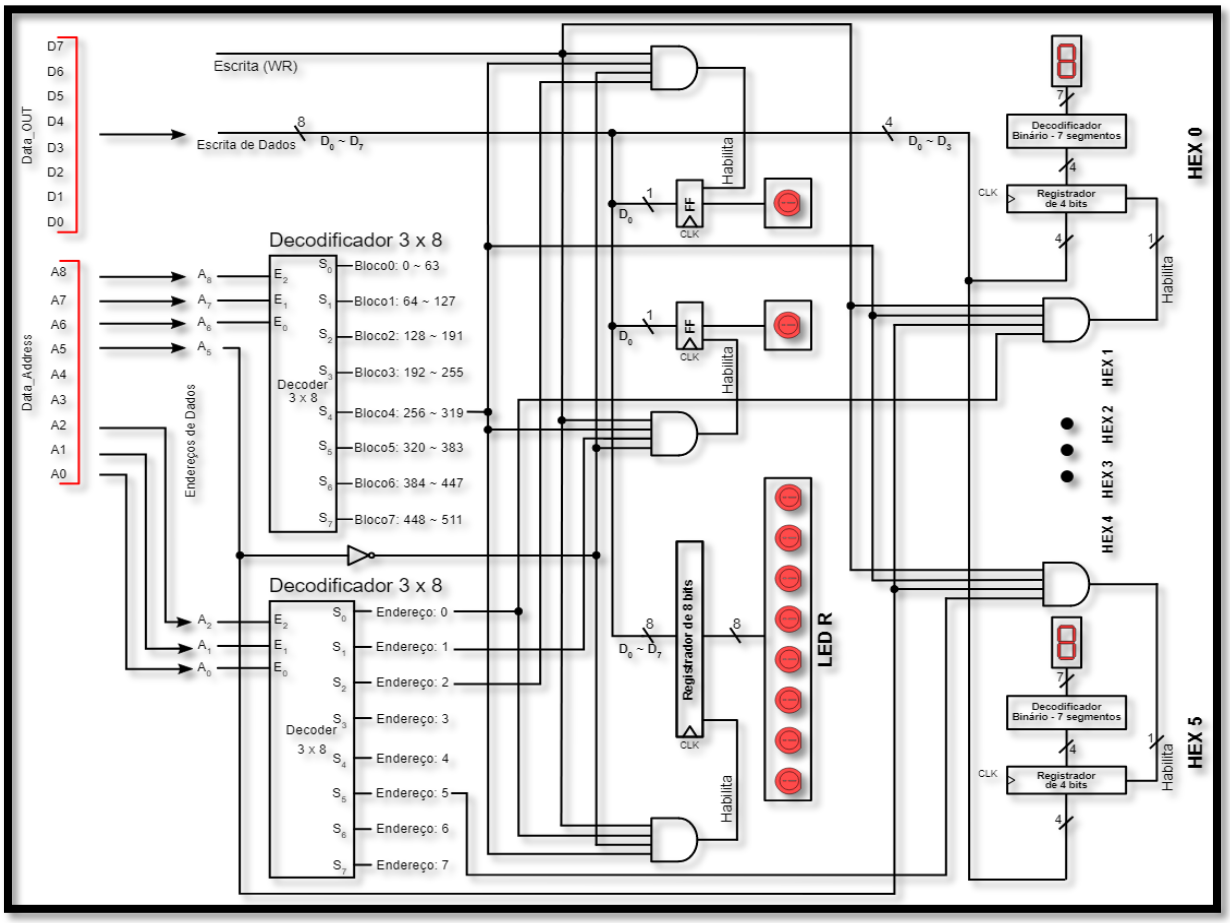

Figura 6: Diagrama retirado do site da matéria feito pelo professor Paulo Carlos Santos. Nele está a lógica de endereçamento de dois periféricos de escrita: os LEDs e o display de sete segmentos.

10) Mapa de Memória

Após a configuração do processador, foi construído um mapa de memória para que seja possível identificar quais endereços foram utilizados em seu endereçamento, listando abaixo todos os endereços relevantes, suas descrições para melhor compreensão e como são acessados.

MAPA DE MEMÓRIA					
End. em decimal	Periférico	Descrição	Largura dos dados	Tipo de acesso	Bloco da memória
0	RAM	Valor da unidade de segundo	8 bits	Leitura/Escrita	0
1	RAM	Valor da dezena de segundo	8 bits	Leitura/Escrita	0
2	RAM	Valor da unidade de minuto	8 bits	Leitura/Escrita	0
3	RAM	Valor da dezena de minuto	8 bits	Leitura/Escrita	0
4	RAM	Valor da unidade de hora	8 bits	Leitura/Escrita	0
5	RAM	Valor da dezena de hora	8 bits	Leitura/Escrita	0
6	RAM	Flag do despertador	8 bits	Leitura/Escrita	0
7	RAM	Constante 0	8 bits	Leitura/Escrita	0
8	RAM	Constante 1	8 bits	Leitura/Escrita	0
9	RAM	Constante 10	8 bits	Leitura/Escrita	0
10	RAM	Valor do despertador para unidade de segundo	8 bits	Leitura/Escrita	0
11	RAM	Valor do despertador para dezena de segundo	8 bits	Leitura/Escrita	0
12	RAM	Valor do despertador para unidade de minuto	8 bits	Leitura/Escrita	0
13	RAM	Valor do despertador para dezena de minuto	8 bits	Leitura/Escrita	0
14	RAM	Valor do despertador para unidade de hora	8 bits	Leitura/Escrita	0
15	RAM	Valor do despertador para dezena de hora	8 bits	Leitura/Escrita	0
16	RAM	Constante 6	8 bits	Leitura/Escrita	0
17	RAM	Constante 2	8 bits	Leitura/Escrita	0
18	RAM	Constante 4	8 bits	Leitura/Escrita	0
19	RAM	Constante 24	8 bits	Leitura/Escrita	0
20˜63	Reservado	-	-	-	0
64˜127	Reservado	-	-	-	1
128˜191	Reservado	-	-	-	2
192˜255	Reservado	-	-	-	3
256	LEDR0˜LEDR7	Endereços dos LEDS 0 ao 7	8 bits	Escrita	4
257	LEDR8	Endereço do LED 8	1 bit	Escrita	4
258	LEDR9	Endereço do LED 9	1 bit	Escrita	4
259˜287	Reservado	-	-	-	4
288	HEX0	Endereço do display 7-seg 0	4 bits	Escrita	4
289	HEX1	Endereço do display 7-seg 1	4 bits	Escrita	4
290	HEX2	Endereço do display 7-seg 2	4 bits	Escrita	4
291	HEX3	Endereço do display 7-seg 3	4 bits	Escrita	4
292	HEX4	Endereço do display 7-seg 4	4 bits	Escrita	4
293	HEX5	Endereço do display 7-seg 5	4 bits	Escrita	4
294˜319	Reservado	-	-	-	4
320	SW0˜SW7	Endereço das chaves 0 a 7	8 bits	Leitura	5
321	SW8	Endereço da chave 8	1 bit	Leitura	5
322	SW9	Endereço da chave 9	1 bit	Leitura	5
323˜351	Reservado	-	-	-	5
352	KEY0	Endereço do botão 0	1 bit	Leitura	5
353	KEY1	Endereço do botão 1	1 bit	Leitura	5
354	KEY2	Endereço do botão 2	1 bit	Leitura	5
355	KEY3	Endereço do botão 3	1 bit	Leitura	5
356	FPGA_RESET	Endereço do botão de reset	1 bit	Leitura	5
357˜383	Reservado	-	-	-	5
384˜447	Reservado	-	-	-	6
448˜510	Reservado	-	-	-	7
510	-	Limpa leitura do botão 1	-	Escrita	7
511	-	Limpa leitura do botão 0	-	Escrita	7

Figura 7: Mapa de memória do contador com o endereço em decimal e o bloco de memória que está referenciando, o periférico ao qual está contido, a descrição, a largura dos dados e o tipo de acesso.

11) Fonte do programa em assembly

O programa em *assembly* está anexado na pasta de entrega como ASM.txt, e esse *assembly* foi convertido para o formato da instrução da memória ROM através do *assembler* (também anexado na pasta de entrega) que foi implementado pelo grupo. Esse *assembler* teve como base o código de Marco Mello retirado do repositório do git a seguir: https://github.com/Insper/DesignComputadores/tree/master/AssemblerASM_BIN_VHDL .

No *assembler* implementado, o formato das instruções era o seguinte:

Instrução @posição da memória ou $imediato, registrador

Esse formato só não se aplica quando o comando é NOP, RET ou algum desvio condicional ou incondicional. No caso dos desvios, há apenas a instrução e a posição

da linha para onde o desvio deveria ser feito, o que pode ser um número ou um label. A diferença entre a entrega passada e a atual está apenas na adição de novas instruções ao dicionário de opcodes do assembler.

12) Referências

[1] SANTOS C. Paulo – Aula 6 – Acesso disponível em <Aula 6>;

[2] SANTOS C. Paulo – Aula 7 – Acesso disponível em <Aula 7>;

[3] SANTOS C. Paulo – Aula 8 – Acesso disponível em <Aula 8>;

[4] SANTOS C. Paulo – Aula 9 – Acesso disponível em <Aula 9>;

[5] SANTOS C. Paulo – Aula 10 – Acesso disponível em <Aula 10>.

[6] MAKUTA S. Lívia; FREZZATTI M. Henrique; QUEIROGA M. Nicolas – P1 – Entrega Intermediária – Contador.