Nome: Antonio da Ressurreição Filho.

Aluno do segundo período do curso de Ciência da Computação da Universidade Federal do Paraná (UFPR).

Professor Doutor Giovanni Venâncio - Projetos Digitais.

Comecei meu trabalho implementando a Unidade Lógica Aritmética (ULA), em que fiz primeiramente o meu somador feito com Full-Adders e Half-Adders. Depois, fiz as operações lógicas AND, OR, XOR e a multiplicação, as quais não exigiram um esforço tão grande pois aproveitei os circuitos já prontos contidos no Digital, segundo recomendação do professor Doutor Giovanni Venâncio. Por fim, fiz o meu subtrador inspirado no meu somador, somente fazendo a mais o circuito do complemento de 2 no componente de 32 bits "B" para que dê certo. O meu deslocador eu utilizei a recomendação do professor dada em sala de aula, inspirando-se nos 5 bits menos significativos e mudando-os conforme pedido, com a ajuda de multiplexadores e shifters de 1,2,4,8 e 16 bits. Detalhe: Eu mesmo fiz o multiplexador da minha ULA, pois queria aprender a fazer um e estava com vontade de deixar o meu circuito de uma forma mais apresentável e menos confuso para qualquer um que fosse vê-lo.

Após minha ULA, comecei a implementação do meu banco de registradores, o qual eu fiz a minha entrada W enable de 1 bit, que seria basicamente meu botão de ativação ligado a um desmultiplexador em que terá a entrada C que é o registrador em que eu escrevo, que manda o sinal para meu banco. Dentro do banco de registradores, há 16 registradores de 32 bits cada, em que tem entradas D (o que vou colocar neles), entrada de Clock (como o nome mesmo já diz, para o Clock) e entrada W (botão de ativação que será transmitido do desmultiplexador qual registrador será ativado). Há 1 saída em cada um deles que será ligada a duas saídas gerais, as quais são "A" e "B", que são saídas de leituras. Por conseguinte, todos esses sistemas se ligam e fazem um banco de registradores, o qual serve de forma fundamental no meu programa principal.

Por fim, implementei meu programa principal, o qual usei essas duas estruturas criadas anteriormente. Nele, tive que implementar alguns circuitos a mais para que realmente funcionasse, como o chamado por mim "registrador do main" que consiste em um somador que irá somar o próximo PC, o próprio PC que usei um registrador de 32 bits anteriormente criado no banco de registradores, extensor de zero e extensor de sinal.

O mais importante e difícil desse trabalho foi fazer as memórias ROM, que são baseadas nos multiplexadores que ativam certa entrada e desativam outra no programa principal e no meu código assembly que eu implementei a partir da tradução do código C dado para se fazer o trabalho. Encontrei muitos desafios durante essa

parte, porém, após algumas horas “desbugando” alguns bugs que apareceram eu consegui implementar todo o trabalho de um micro-processador inspirado no MIPS com algumas modificações.

Código Assembly:

```
1       start
2       lw $t8, N(10)
3       lw $t9, a(7)
4       lw $t10, d(18)
5       lw $t11, soma(0)
6       lw $t12, i(0)
7       loop_start: beq $t12,$t8,loop_end(7)
8           mul $t13, $t12, $t10
9           add $t14, $t13, $t9
10          add $t11, $t11, $t14
11          show $t11
12          addi $t12, $t12, 1
13      j loop_start(-7)
14  loop_end: show $t11
15      finaliza
```

Binário:

0111 0000 0000 0000 0000000000000000

0001 0000 0000 1000 0000000000001010

0001 0000 0000 1001 0000000000000111

0001 0000 0000 1010 0000000000010010

0001 0000 0000 1011 0000000000000000

0001 0000 0000 1100 0000000000000000

1100 1100 1000 0000 0000000000000111

0100 1100 1010 1101 0000000000000000

0010 1101 1001 1110 0000000000000000

0010 1011 1110 1011 0000000000000000

1101 1011 0000 0000 0000000000000000

1001 1100 0000 1100 0000000000000001

1011 0000 0000 0000 1111111111111001

1101 1011 0000 0000 0000000000000000

1111 0000 0000 0000 0000000000000000

Hexadecimal:

70000000

1008000A

10090007

100A0012

100B0000

100C0000

100D0001

CC800007

4CAD0000

2D9E0000

2BEB0000

DB000000

9C0C0001

B000FFF8

DB000000

F0000000