Trabalho Prático de Projetos Digitais e Microprocessadores

Guilherme Vitoriano Santana de Oliveira GRR:20245396

Componentes desenvolvidos para o trabalho:

- Somador 1 bits
  Componente usado para facilitar a confecção do somador de 2 bits
- Somador 2 bits
  Componente usado para facilitar a confecção do somador de 8 bits
- Somador 8 bits
  Componente usado para facilitar a confecção do somador de 32 bits
- Somador 32 bits
  Componente usado dentro da ULA e para o cálculo do próximo PC.
- Subtrator 32 bits
  Componente usado dentro da ULA
- Shifter 32 bits
  Componente usado dentro da ULA
- ULA
  Componente que recebe dados armazenados em 2 registradores, ou no imediato e realiza alguma manipulação antes de enviá-lo de volta para o Banco de Registradores
- Registrador 32 bits
  Armazena dados para uso futuro
- Banco de Registradores
  Conjunto de 16 registradores de 32 bits
- Extensor de sinal
  Usado para transformar o imediato de 16 bits para 32 bits mantendo seu valor
- Instrução
  Utilizado para separar bits da instrução e enviá-los ao lugar onde serão utilizados
- Sinal de Controle
  Separa os bits da Memória de controle e os envia para onde serão usados, possui 9 bits,
  0:Write enable
  1:Branch
  2:ULA imediato
  3-5: Operação da ula
  6:Display
  7:Halt
  8:Jump
- Bloco operativo
  Engloba todos os componentes e circuitos do projeto.

Memórias:

Memória de Instrução:

Onde o código binário está armazenado, o linha de código a ser executada é definida pelo Registrador PC

Memória de Controle:

Recebe o OPcode da função e de acordo com o OPcode envia um sinal de 9 bits para o Sinal de controle

Circuito:

O registrador PC define qual instrução deve ser executada pela memória de instruções, que então propaga 32 bits de dados para o circuito que são divididos

em 5 segmentos, os 4 bits mais significativos são o OPcode, os 4 seguintes definem o de qual registrador será uma das leituras, os próximos 4 definem o segundo registrador da leitura, os 4 bits em seguida apontam para qual registrador receberá os dados e por fim os últimos 16 bits são o imediato.

Em relação aos desvios de fluxo, decidi que todos os desvios seriam calculados a partir do pc+1.

Escolhi fazer os sinais de controle de maneira que não importa o que está nos bits não utilizados da instrução, o que aumentou a quantidade de sinais de controle consideravelmente, mas também deixa o circuito mais robusto caso haja erros no código.

Implementação em assembly:

```
        li r1,10
        li r2,7
        li r3,18
        add r4, zero, zero
        add r5, zero, zero
loop:
      beq r5, r1, fim_loop
      mul r6, r5, r3
      add r6, r2, r6
      add r4, r4, r6
      show r4
      addi r5, r5, 1
      j loop
fim_loop:
      show r4
      halt
```

No código em assembly decidi usar “add” no r4 e no r5 ao invés de “li” devido ao fato do número carregado ser 0, então como o r0 sempre é zero então é mais rápido usá-lo do que um imediato que precisaria passar na extensão de sinal.

Código em binário:

Opcode	A	B	C	Imm
1001	0000	0000	0001	00000000 00001010
1001	0000	0000	0010	00000000 00000111
1001	0000	0000	0011	00000000 00010010
0010	0000	0000	0100	00000000 00000000
0010	0000	0000	0101	00000000 00000000
1100	0101	0001	0000	00000000 00000111
0100	0101	0011	0110	00000000 00000000
0010	0110	0010	0110	00000000 00000000
0010	0110	0100	0100	00000000 00000000
1101	0100	0000	0000	00000000 00000000
1001	0101	0000	0101	00000000 00000001
1011	0000	0000	0000	11111111 1111001
1101	0100	0000	0000	00000000 00000000
1110	0000	0000	0000	00000000 00000000