Relatório - Trabalho de Projetos Digitais

Nome: Alejandro David Nava Nava.
GRR: 20242778.

Detalhes da implementação:

1. Em alguns casos, foram feitos circuitos menores que posteriormente foram combinados para atingir os 32 bits (ex.: somador/subtrator e shifter);
2. O subtrator da *ULA* reutiliza o circuito do somador, apenas invertendo os bits da entrada B e colocando o primeiro *Carry in* como 1;
3. A codificação nas memórias de instruções e controle foi feita em hexadecimal por limitações do próprio simulador;
4. Os 8 bits mais significativos do PC são desconsiderados ao entrar na *memória de instruções* pois do componente *Memória ROM* do simulador só admite 24 bits de entrada;
5. Foi configurado um *opcode* diferente para cada instrução.

Sinais de controle:

No circuito, existem 9 bits que representam sinais de controle armazenados na *memória de controle* e são decodificados a partir do *opcode:*

- **Aluop** (3 bits): indicam a operação a ser feita pela *ULA*:
- **Dis cont** (1 bit): habilita a atualização do Display;
- **Alu src** (1 bit): se for 1 a *ULA* receberá um imediato como segundo operando, se for 0, receberá o conteúdo de um registrador;
- **W src** (1 bit): se for 1, no Banco de registradores será escrito um imediato, se for 0, o que será escrito vai vir da *ULA*;
- **W en** (1 bit): permite ou não a escrita no *B.R*;
- **J** (1 bit): determina se deve ocorrer um *jump;*
- **B** (1 bit): determina se deve ocorrer um branch, caso a condição for verdadeira.

Op.	Aluop
add	000
sub	001
mult	010
and	011
or	100
xor	101
sll	110

Operação	Op code	Mem. de controle				
nop	0000	000000000		xor	0111	001000101
li	0001	001100000		sll	1000	001000110
add	0010	001000000		addi	1001	001010000
sub	0011	001000001		subi	1010	001010001
mult	0100	001000010		jump	1011	010000000
and	0101	001000011		beq	1100	100000001
or	0110	001000100		show	1101	000001000
				halt	1110	010000000

Código em Assembly a partir do programa de C disponibilizado:

```
        li r1, 0        ; registrador que acumula a soma da PA
        li, r2, 7       ; termo inicial
        li r3, 18       ; razão
        li r4, 10       ; número de termos
        li r5, 0        ; iterador
loop:   beq r4, r5, fim
        mult r6, r5, r3
        add r1, r1, r6
        add r1, r1, r2
        addi r5, r5, 1
        show r1
        j loop
fim:    show r1
        halt
```

Formato das instruções:

Apesar de cada uma das instruções terem *opcodes diferentes,* seguem o seguinte padrão:

	32 bits				
	4 bits	4 bits	4 bits	4 bits	16 bits
Tipo r	opcode	rd	rs1	rs2	x
Tipo i	op code	rd	rs1	x	imediato
Jump	op code	x	x	x	deslocamento (imediato)
Branch	op code	x	rs1	rs2	deslocamento (imediato)
li	op code	rd	x	x	imediato
show	op code	x	rs1	x	x

- **rd**: registrador de destino;
- **rs1**: registrador de origem 1;
- **rs2**: registrador de origem 2;
- **x**: don't care (bits que não interferem na instrução).