Relógio Documentação

Autores:

- José Hélio Paiva Neto *(Usuário no git: heliopn)*
- Pedro Vero Fontes *(Usuário no git: fontes99)*
- Rafael Almada *(Usuário no git: slimkaki)*

Link do repositório:

https://github.com/slimkaki/RelogioDescomp

Sobre o projeto

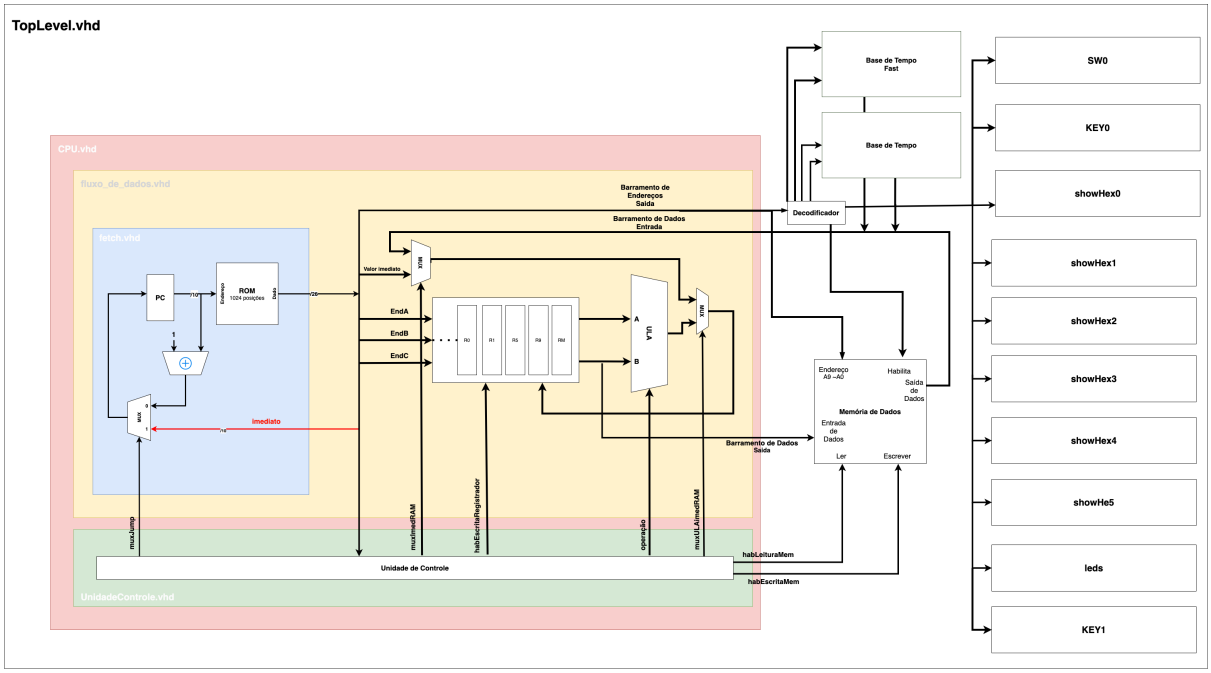
O projeto consiste em escolher uma arquitetura de processador e implementá-la, de forma que ao fim teremos um relógio funcional na FPGA, com algumas funcionalidades.

Arquitetura

> Registrador - Registrador

Com uma arquitetura do tipo, iremos trabalhar com instruções que usam apenas registradores.

O esquema da arquitetura, conforme mostrada em aula, é o seguinte:



Instruções

Foi pensado em um máximo de 16 instruções, estas que são representadas pelos 4 bits iniciais de cada instrução entregue ao processador.

Além disso, foi optado por utilizar 16 registradores e, dependendo da instrução, a disponibilidade de usar um número entre 1 e 3 registradores.

Por fim foram adicionados 10 bits no fim de cada instrução que são destinados puramente para enviar valores para os registradores, como na instrução lea.

```
23   19   15   11   7
0000 0000 0000 0000 00000000   → 26 bits
  20   16   12    8        0
|    |    |    |    |
1    2    3    4    5
```

1. Opcode (Max de instruções = 16)
2. Endereço Registrador A (gravação)
3. Endereço Registrador B
4. Endereço Registrador C (leitura)
5. Imediato (8 bits)

1. **lea**
    - **Nome:** Load Effective Address
    - **OpCode da instrução:** 0000
    - **Exemplo:** lea $RA, "7"
    - **Descrição:** Salva o valor representado por 7 no registrador *RA*.
    - **Instrução decodificada:** 0000 *0001* 0000 0000 00000111
2. **mov**
    - **Nome:** Move
    - **OpCode da instrução:** 0001
    - **Exemplo:** mov $RA, $RC
    - **Descrição:** Move (copia) o dado do registrador *RC* para o registrador *RA*
    - **Instrução decodificada:** 0001 *0001* 0000 *0010* 00000000
3. **add**
    - **Nome:** Addition
    - **OpCode da instrução:** 0010
    - **Exemplo:** add $RA, $RB, $RC

- **Descrição:** Operação de adição do registrador *RB* com o registrador *RC* e salva no *RA*
- **Instrução decodificada:** 0010 *0001 0010 0011* 00000000

4. **sub**
    - **Nome:** Subtraction
    - **OpCode da instrução:** 0011
    - **Exemplo:** sub $R1, $R2, $R3
    - **Descrição:** Operação de subtração do registrador *RB* com o registrador *RC* e salva no *RA*
    - **Instrução decodificada:** 0011 *0001 0010 0011* 00000000
5. **inc**
    - **Nome:** Increment
    - **OpCode da instrução:** 0100
    - **Exemplo:** inc $RA
    - **Descrição:** Operação de incremento (soma 1) no registrador *RA*
    - **Instrução decodificada:** 0100 *0001* 0000 *0001* 00000000
6. **je**
    - **Nome:** Jump if equal
    - **OpCode da instrução:** 0101
    - **Exemplo:** je $RC, $RB, :label
    - **Descrição:** Instrução jump para o endereço '0xEnd' se o dado no *RC* for igual ao dado no *RB*
    - **Instrução decodificada:** 0101 0000 *0010 0001* 00100110
7. **jl**
    - **Nome:** Jump if less than
    - **OpCode da instrução:** 0110
    - **Exemplo:** jl $RC, $RB, :label

- **Descrição:** Instrução jump para o endereço '0xEnd' se o dado no *RC* for menor que o dado no *RB*
- **Instrução decodificada:** 0110 0000 *0010 0001* 00000111

8. **jle**
    - **Nome:** Jump if less or equal than
    - **OpCode da instrução:** 0111
    - **Exemplo:** jle $RC, $RB, :label
    - **Descrição:** Instrução jump para o endereço '0xEnd' se o dado no *RC* for menor igual ao dado no *RB*
    - **Instrução decodificada:** 0111 0000 *0010 0001* 10100110
9. **jmp**
    - **Nome:** Jump
    - **OpCode da instrução:** 1000
    - **Exemplo:** jmp, :label
    - **Descrição:** Apenas jump para o endereço '0xEnd'
    - **Instrução decodificada:** 1000 0000 0000 0000 00100111
0. **nop**
    - **Nome:** No operation
    - **OpCode da instrução:** 1001
    - **Exemplo:** nop
    - **Descrição:** Linha de não operação
    - **Instrução decodificada:** 1001 0000 0000 0000 00000000
1. **load**
    - **Nome:** Load
    - **OpCode da instrução:** 1011
    - **Exemplo:** load $RA, 0xEnd
    - **Descrição:** Carrega em *RA* o valor da memoria com endereço 0xEnd

- **Instrução decodificada**: 1011 0101 0000 0000 00000111

2. **store**
    - **Nome**: Store
    - **OpCode da instrução**: 1100
    - **Exemplo**: store $RB, 0xEnd
    - **Descrição**: Guarda o valor de *RB* na memória, no local da memoria com 0xEnd
    - **Instrução decodificada**: 1100 0000 0111 0000 00000100

Nossos registradores:

1. **segU**
    - Usado para registrar o valor da unidade dos segundos que aparece no display de sete segmentos.
2. **segD**
    - Usado para registrar o valor da dezena dos segundos que aparece no display de sete segmentos.
3. **minU**
    - Usado para registrar o valor da unidade dos minutos que aparece no display de sete segmentos.
4. **minD**
    - Usado para registrar o valor da dezena dos minutos que aparece no display de sete segmentos.
5. **horU**
    - Usado para registrar o valor da unidade das horas que aparece no display de sete segmentos.
6. **horD**
    - Usado para registrar o valor da dezena das horas que aparece no display de sete segmentos.
7. **R0**

- Registrador utilizado para guardar o valor zero.

8. **R1**
    - Registrador utilizado para guardar o valor um.
9. **R2**
    - Registrador utilizado para guardar o valor dois.
0. **R9**
    - Registrador utilizado para guardar o valor nove.
1. **R5**
    - Registrador utilizado para guardar o valor cinco.
2. **RM**
    - Registrador utilizado para guardar valores de memória.
3. **RT**
    - Registrador utilizado para a base temporal.

Para rodar o assembler presente na pasta assembler

```
$ python3 assembler.py
```

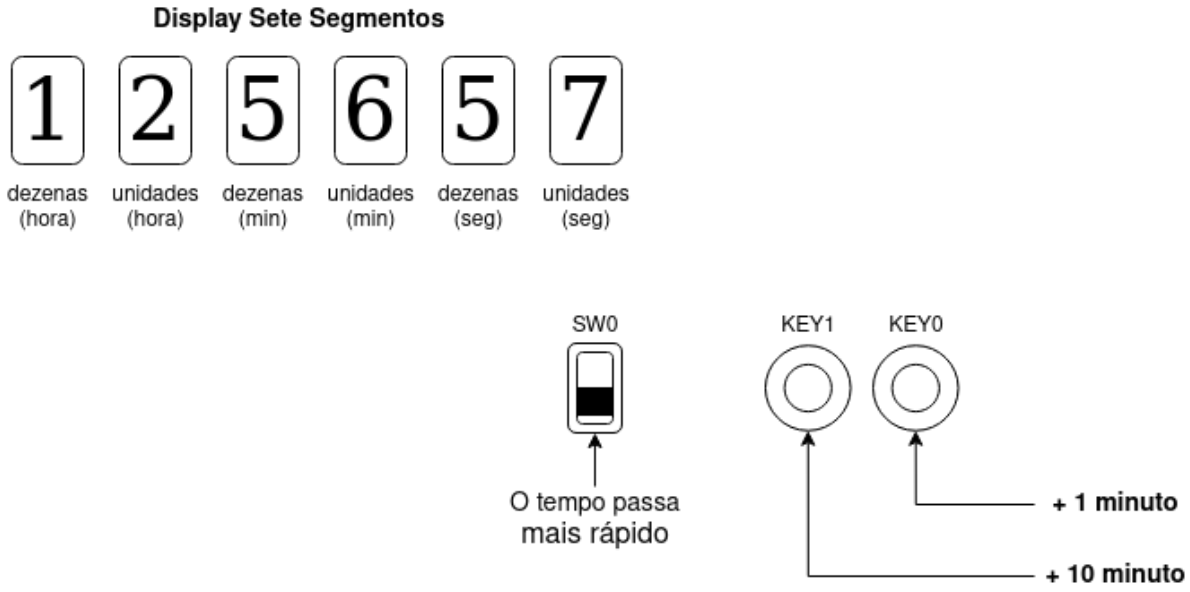
Assim que rodar, é gerado um código "initROM.mif" que está sendo importado no vhdl **ou** então também é possível substituir as linhas que começam com "tmp()" pelas linhas do arquivo "assembled_<nome do arquivo>.txt" dentro do arquivo memoriaRom.vhd

P.S.: É esperado que todos os jumps sejam direcionados à uma label dentro do código assembly.

Interação com o usuário

Além da funcionalidade principal de mostrar ao usuário o horário atual, pretendemos colocar algumas coisas a mais:

Dessa forma temos:



1. **SW0**
    - O tempo passa mais rápido
2. **KEY0**
    - Incrementa 1 minuto a mais no relógio
3. **KEY1**
    - Incrementa 10 minuto a mais no relógio