# Pontifícia Universidade Católica de Campinas

Raissa Furlan Davinha RA: 15032006

Rafael Fioramonte RA:16032708

Bruno Vicente Donaio Kitaka RA:16156341

**CPU Simples em VHDL**

Arquitetura de Computadores

Campinas

2017

**INTRODUÇÃO**

Em aula foi lançado o projeto para criação de uma CPU com 4 registradores de 8 bits cada, que executa 4 instruções simples, de mover, trocar, somar e subtrair. O processador foi feito em linguagem VHDL pelo software Quartus II 13.1.

VHDL é uma linguagem usada para facilitar o design (projeto/concepção) de circuitos digitais em CPLDs, [FPGAs](https://pt.wikipedia.org/wiki/FPGA) e [ASICs](https://pt.wikipedia.org/wiki/ASIC).

Altera Quartus II é um software de design de [dispositivo de lógica programável](https://en.wikipedia.org/wiki/Programmable_logic_device) produzido pela [Altera](https://en.wikipedia.org/wiki/Altera) . Quartus II permite a análise e síntese de projetos de V[HDL](https://en.wikipedia.org/wiki/Hardware_description_language) , que permite ao desenvolvedor compilar seus projetos, realizar análise de tempo, examinar diagramas, simular a reação de um projeto a diferentes estímulos e configurar o dispositivo alvo com o programador.

**DESENVOLVIMENTO**

A CPU é constituida basicamente por um banco de registradores, com 4 registradores de 8bits cada, uma ULA que executa somas, subtrações ou apenas passa adiante o valor do registrador, e uma unidade de controle. Dentro dessasseções principais, foram também utilizados multiplexadores, decodificadores 2 para 4, para seleção das saídas dos registradores, tristate buffers, e flip flops.

Os registradores são identificados por dois bits, totalizando as quatro opões de registradores.

O formato das instruções, e o OPCODE, tendo um total de 8 bits, foi dividido em dois formatos distintos, um para o tipo R, com movimento apenas entre os registradores, e um tipo I, que faz uso de um valor inteiro, chamado de imediado.

Tipo R:

Cada espaço da instrução possui tamanho de 2 bits.

|  |  |  |  |
| --- | --- | --- | --- |
| OPCODE | REGISTRADOR 1 | REGISTRADOR 2 | FUNCTION |

Tipo I:

O espaço opcode e registrador 1 possuem tamanho de 2 bits, enquanto que o imediato possui tamanho igual a 4 bits, chegando ao valor 1111 binário.

|  |  |  |
| --- | --- | --- |
| OPCODE | REGISTRADOR 1 | IMEDIATO |

No tipo I, os números representados não são sinalizados, pois existe a instrução de subtração com números imediatos.

**Tabela de sinais de controle**

|  |  |  |
| --- | --- | --- |
| Instrucoes R | OPCODE | FUNCTION |
| XCHG | 00 | 00 |
| MOV | 00 | 01 |
| ADD | 00 | 10 |
| SUB | 00 | 11 |

|  |  |  |
| --- | --- | --- |
| Intrucoes I | OPCODE | FUNCTION |
| MOV | 01 | XX |
| ADDI | 10 | XX |
| SUB | 11 | XX |

A unidade de controle, recebe os sinais acima, e gera os seguintes sinais, dependendo da instrução:

Xchg (controla dois multiplexadores, antes e depois do banco de registradores, para permitir a entrada dos dados aos registradores corretos);

RegWrite (libera ou não a entrada de dados ao registrador indicado);

TempWrite (sinal para escrita no registrador temporário).

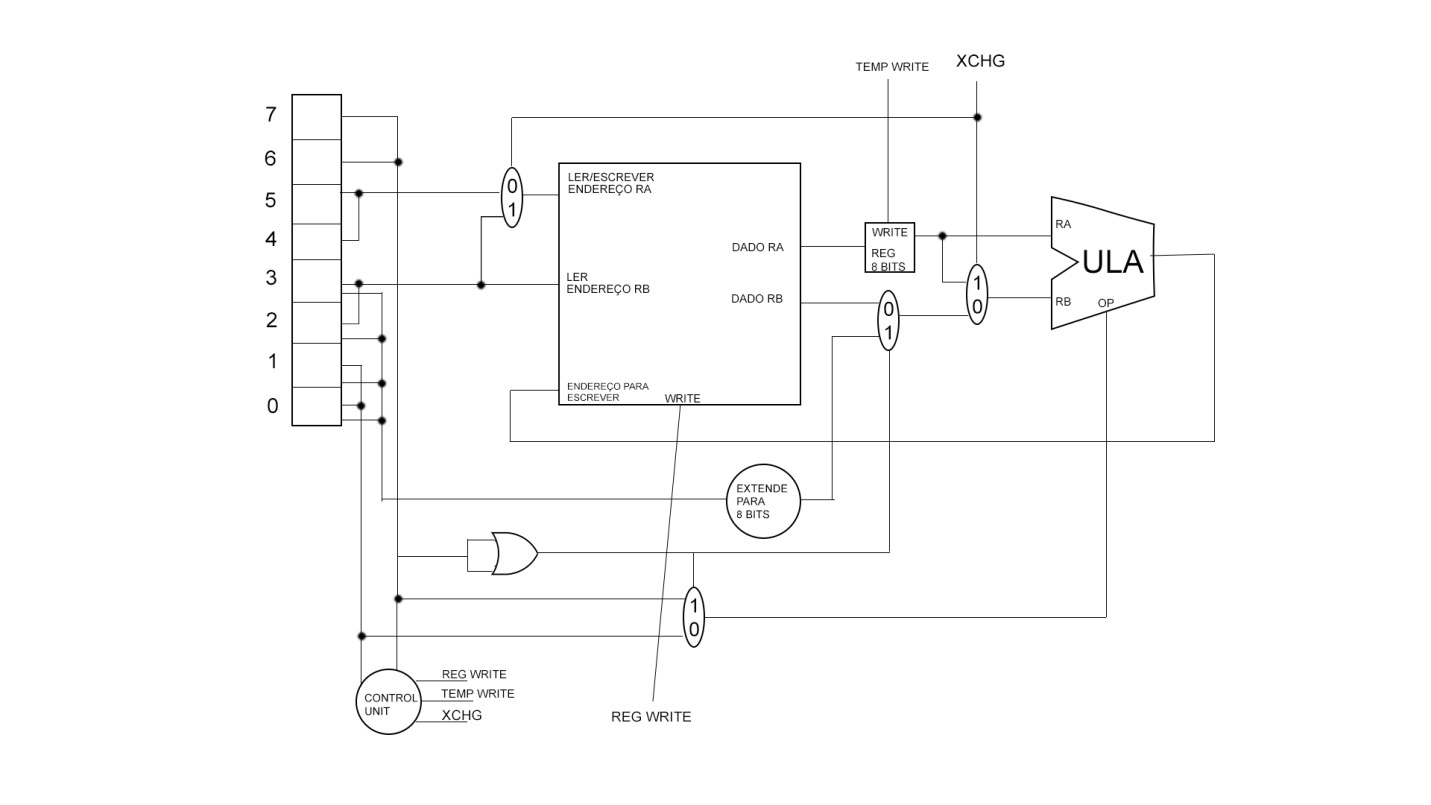
Todas as intruções executam em apenas dois ciclo de clock, exceto a instrução exchange, que necessita de um terceiro ciclo para completar.

Imagem 1: A CPU.

Para selecionar se a segunda entrada da unidade lógica será a saída do registrador RB ou um imediato, e a entrada de controle da mesma será o próprio opcode ou o campo function, dois multiplexadores são controlados pelo opcode. Caso haja o número “1” em qualquer posição do opcode, testado por um OR, a instrução será do tipo Ie o controle da ULA será os 2 bits do opcode. Se não houver o número “1” em qualquer posição, os multiplexadores selecionarão as entradas 0, passando a saída do resgistrador RB a ULA e o campo function para controle do cálculo.

Para controle da unidade logica (ULA), caso o campo OP seja 00, quem entra e o FUNC, caso o campo OP seja diferente de 00, quem entra e o proprio OP.

A entrada de 2 bits da ULA separados em dois caminhos, definem por meio de um multiplexador, se a saída será o resultado da conta feita, ou se apenas passará os dados do segundo registrador para frente, e se será feito uma soma ou subtração com os dados recebidos, reusltando em uma saída de 8 bits.

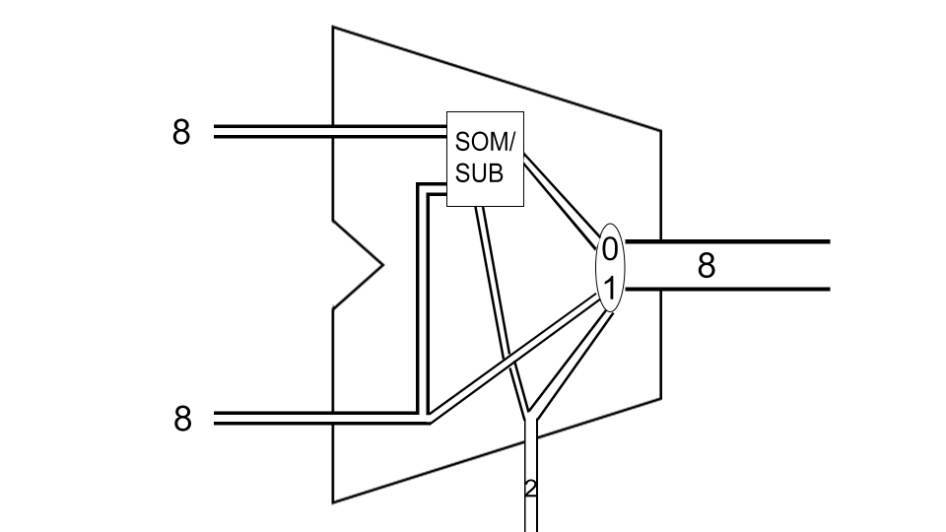


Imagem 2: A ULA.

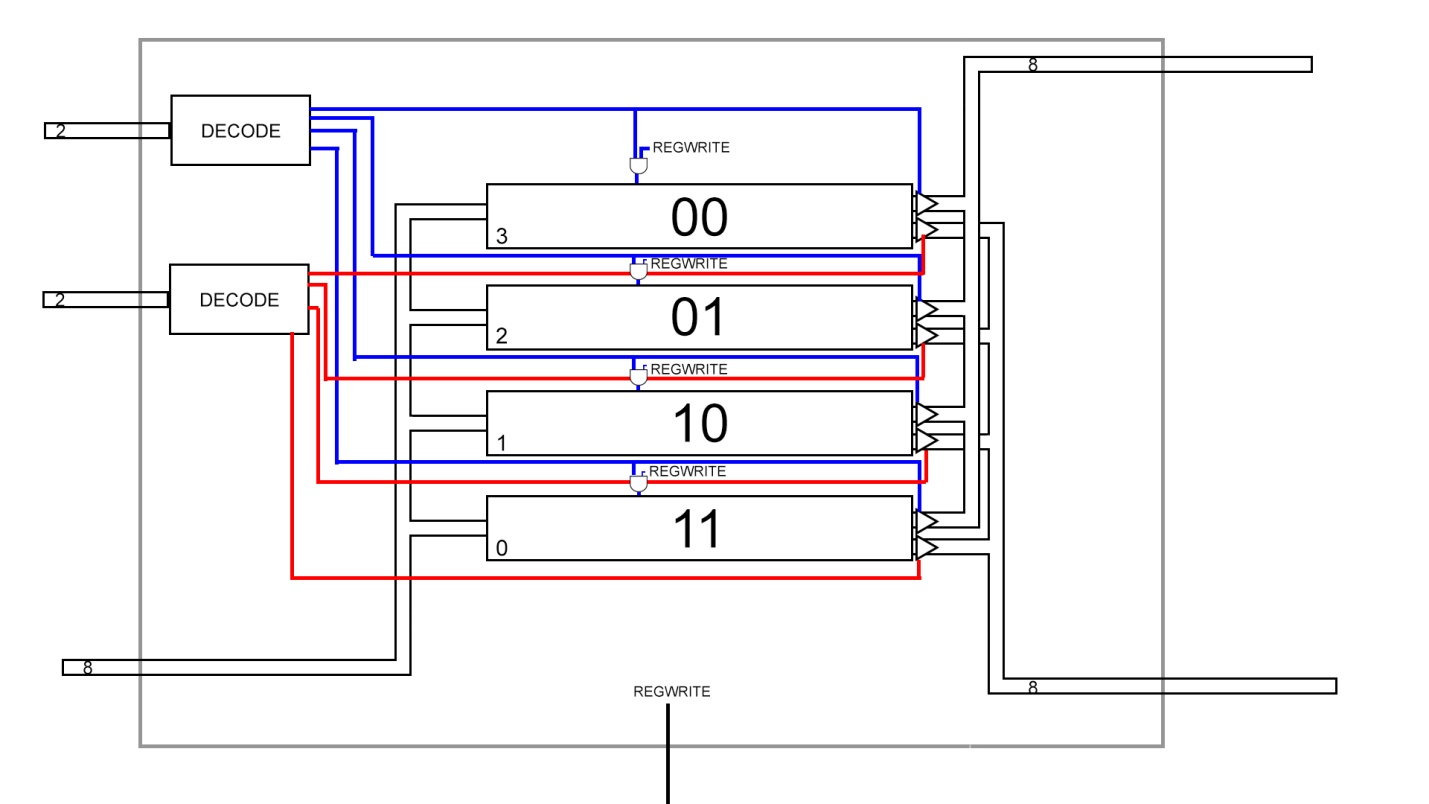


Imagem 3: Os registradores

**Testes realizados**

O circuito foi testado através da própria ferramenta contida no programa Quartus, onde simulamos entradas no formato do opcode criado, e vemos os sinais gerados na saída do circuito.

**Soma**

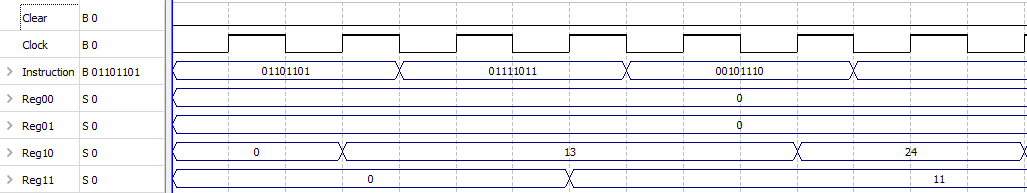
A soma de dois registradores, no caso damos o comando MOVI do imediato 13 para o registrador Reg10, então movemos o imediato 11 para o Reg11 e então somamos ambos registradores.

Figura 4: O teste da soma.

**Soma Com Imediato**

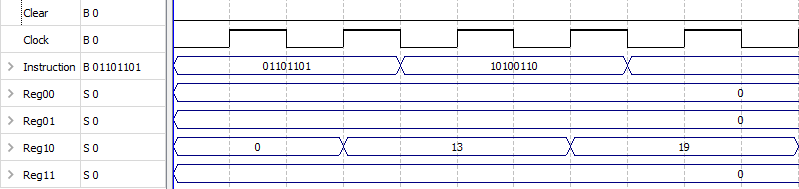
A soma de um registrador com um imediato, no caso movemos o imediato 13 para o registrador Reg10 e então somamos ele com o imediato 6.

Figura 5: O teste da soma com imediato.

**MOV**

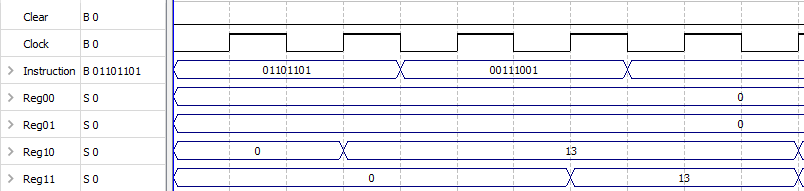
A movimentação do conteúdo de um registrador para outro, no caso movemos o imediato 13 para o registrador Reg10 e então um MOV de Reg10 para o Reg11.

Figura 6: O teste do MOV.

**MOVI**

A movimentação do conteúdo de um imediato para um registrador, no caso movemos o imediato 13 para o registrador Reg10.

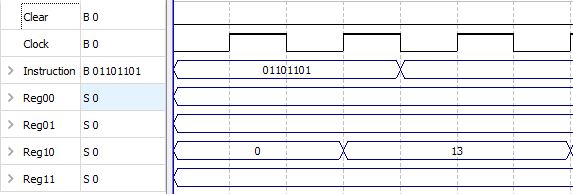


Figura 6: O teste do MOVI.

**Subtração**

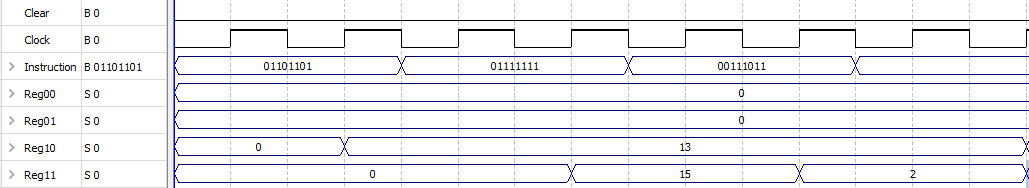
 A subtração de dois registradores, no caso movemos o imediato 13 para o registrador Reg10, então movemos o imediato 15 para o Reg11 e então subtraímos Reg10 de Reg11.

Figura 7: O teste da subtração.

**Soma Com Imediato**

A subtração de um registrador com um imediato, no caso movemos o imediato 13 para o registrador Reg10 e então subtraímos dele o imediato 14.

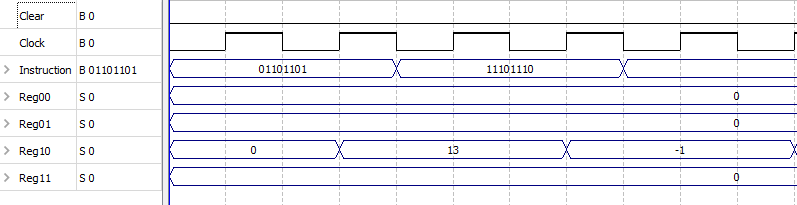
****

Figura 8: O teste da subtração com imediato.

**XCHG**

A troca dos conteúdos de dois registradores, no caso movemos o imediato 13 para o registrador Reg10 e o imediato 15 para o Reg11, então damos o comando XCHG e o conteúdo dos dois registradores são trocados entre si.

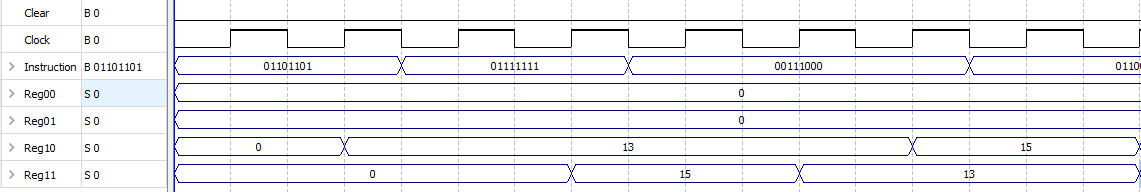
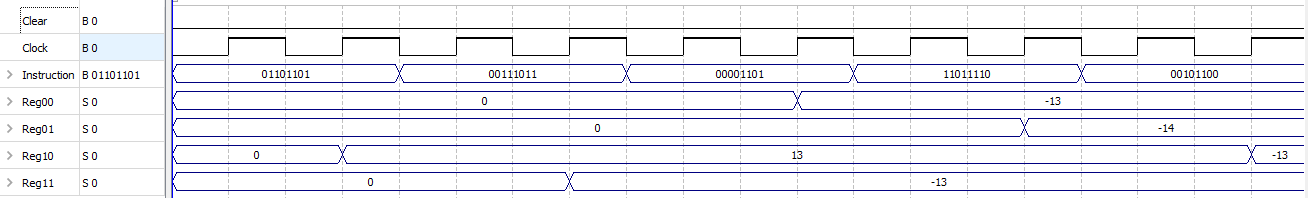
****

Figura 9: O teste do XCHG.

**Teste Geral**

Movemos o imediato 13 para Reg10, subtraímos o Reg10 do Reg11, movemos o conteúdo de Reg11 para Reg00, subtraímos o imediato 14 do Reg01 e então fazemos um exchange entre Reg10 e Reg11. 

**Referências**

VHDL. Disponível em: <https://pt.wikipedia.org/wiki/VHDL>.

Altera Quartus. Disponível em: <https://en.wikipedia.org/wiki/Altera\_Quartus>.

Binary to Decimal to Hexademical Converter. Disponível em: <https://www.mathsisfun.com/binary-decimal-hexadecimal-converter.html>.