Manual do MIPS Single-Cycle

Grupo: Antonio Fuziy, André Tavernaro e Marcelo Cesário

Professores: Paulo Santos e Marco Alves

Design de Computadores

Instruções:

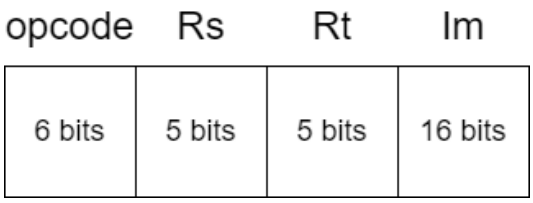
Nesse tópico vale identificar todos os tipos de instruções que o MIPS Single-Cycle da entrega intermediária consegue realizar, essas foram mostradas na tabela 1 abaixo:

Tabela 1:

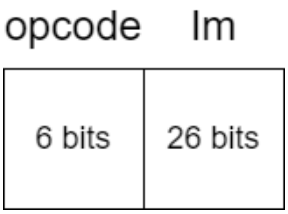
Instrução	OPCODE / FUNCT	Operação	Formato	Operação ULA
LW	**100011** / -	R[rt] = M[R[rs]+SignExtImm]]	**Tipo I**	**010**
SW	**101011** / -	M[R[rs]+SignExtImm] = R[rt]	**Tipo I**	**010**
ADD	000000 / **100000**	R[rd] = R[rs] + R[rt]	**Tipo R**	**010**
SUB	000000 / **100010**	R[rd] = R[rs] - R[rt]	**Tipo R**	**110**
AND	000000 / **100100**	R[rd] = R[rs] & R[rt]	**Tipo R**	**000**
OR	000000 / **100101**	R[rd] = R[rs] | R[rt]	**Tipo R**	**001**
SLT	000000 / **101010**	R[rd] = (R[rs] < R[rt]) ? 1: 0	**Tipo R**	**111**
BEQ	**000100** / -	if(R[rs]==R[rt]) PC=PC+4+BranchAddr	**Tipo I**	**110**
JMP	**000010** / -	PC=JumpAddr	**Tipo J**	**-**

Formato das instruções

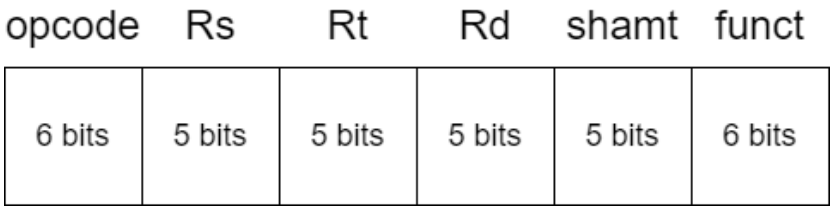
- Instruções do tipo I



- Instruções do tipo J



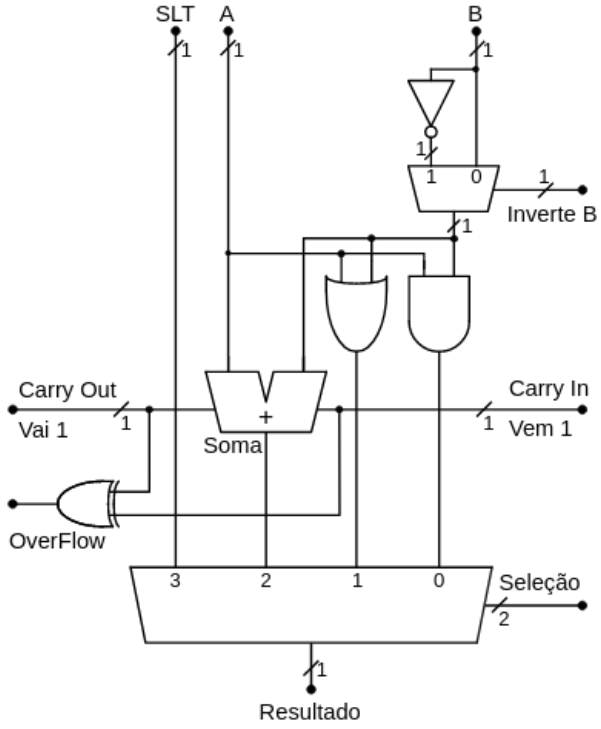
- Instruções do tipo R



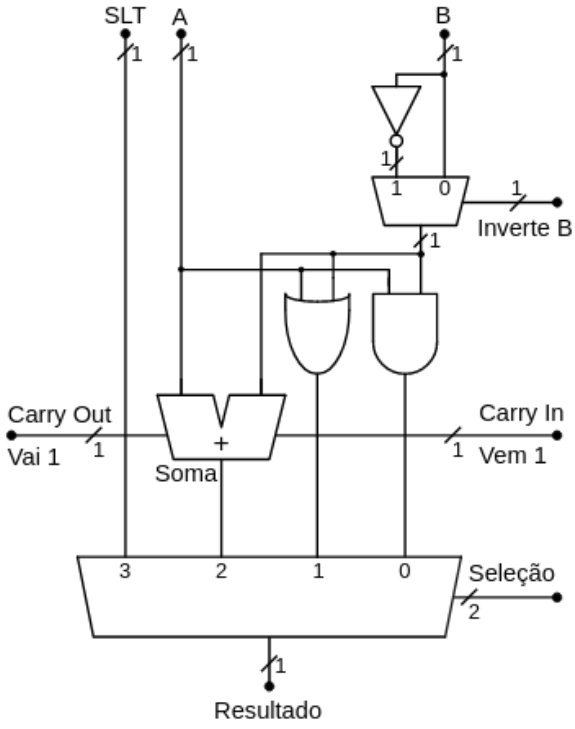
Fluxo de dados do processador

A ULA do nosso processador é composta por 32 ULAs de 1 bit, a conexão delas foi feita de tal modo que a C Out de uma é o C Out da outra. A conexão das ULAs pode ser visto na imagem abaixo:

ULA Overflow:



ULA de 1 bit comum:

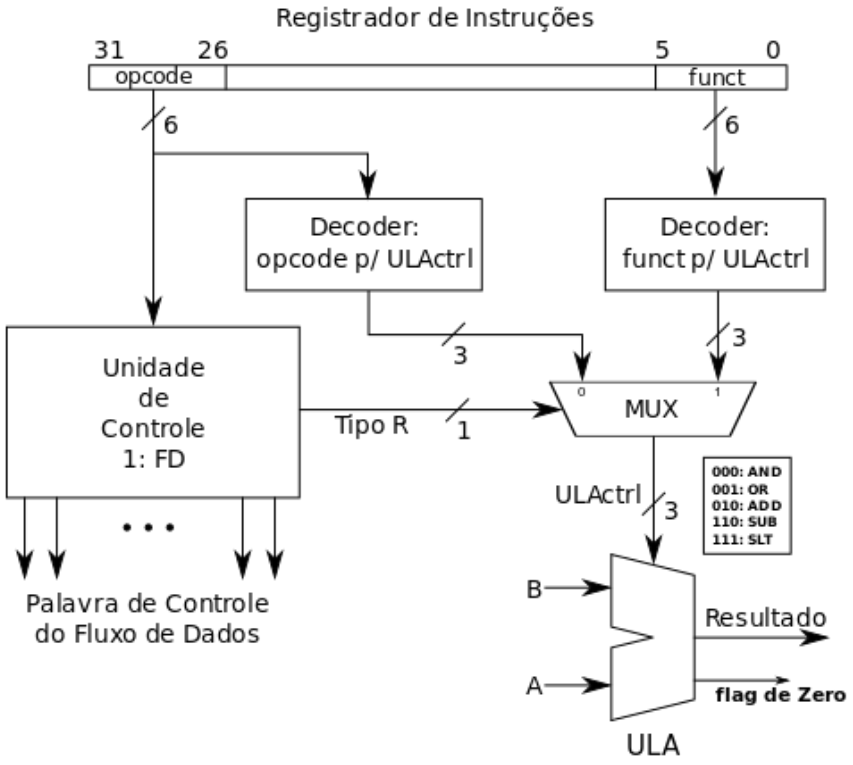


Diante das 32 ULAs, somente duas delas possuem um diferencial em sua configuração:

- Na ULA do último Bit, ela possui uma saída extra, que é responsável por mostrar o sinal de overflow, que é utilizado na instrução SLT. A função desse sinal para o SLT é basicamente dizer se o resultado final foi negativo.
- Na ULA do primeiro bit, como ela não possui uma ULA anterior a ela, e consequentemente, o C In não é o C Out de outra ULA, então o que foi configurado nela é que o C In é o Inverte B, que serve para realizar operações de subtração.

Para controlar e administrar as operações da ULA de 32 bits foi construído uma unidade de controle do fluxo de dados, o grupo optou por escolher a segunda opção de

implementação, utilizando 3 decoders. A configuração dele pode ser visto na imagem abaixo:



A unidade de controle funciona com 3 decoders e um Mux:

- A unidade de controle do fluxo de Dados, uma de suas saídas conectada ao seletor do Mux, caso o tipo de instrução seja tipo R o Mux pega a saída do decoder do FUNCT e caso contrário ele pega a saída do decoder do OPCODE.

- O primeiro decoder cuida das instruções de tipo I, desse modo, ele recebe o OPCODE, e envia a operação de tipo I que a ULA deve realizar dependendo da instrução para a entrada A do MUX.
- O segundo decoder cuida das instruções de tipo R, desse modo, ele recebe o FUNCT, e envia a operação de tipo R que a ULA deve realizar dependendo da instrução para a entrada B do MUX.

Listagem dos pontos de controle do Fluxo de Dados:

- **MUX([PC+4, BEQ] / JMP):** Define se o fluxo seguirá para a próxima instrução ou realizará o jump;
- **MUX(Rt/Rd):** Esse ponto de controle define se a entrada do banco de registradores será o Rt ou o Rd, dependendo do tipo de instrução I ou R;
- **HabEscritaReg:** Esse ponto de controle define se será realizada a escrita de dados no banco de registradores;
- **MUX(Rt/Imediato):** Esse bit do ponto de controle tem por função definir a entrada B da ULA, podendo ser o Imediato ou o valor de Registrador Rt;
- **ULAop:** Esse ponto de controle possui 3 bits, definindo a operação que a ULA irá realizar dependendo da instrução recebida, podendo ser soma, subtração e etc, lembrando que esse ponto de controle é controlado pela Unidade de controle da ULA;
- **BEQ:** Esse ponto de controle define se a instrução do tipo BEQ vai acontecer ou não;

- **habLeituraMEM:** Esse bit do ponto de controle define se será feita a leitura da memória RAM;
- **habEscritaMEM:** Esse bit do ponto de controle define se será feita a escrita da memória RAM.

Manual de Uso:

Na simulação do código do projeto do MIPS Single-Cycle, vale ressaltar alguns pontos sobre as como funcionará a FPGA:

- **SevenSegs e LEDs:** Os displays e LEDs da placa devem mostrar o valor da saída da ULA ou o valor do Program Counter atual, essa condição é controlada pela SW(0);
- **SW(0):** A switch 0 define qual será o valor mostrado nos displays, caso esteja ativa, o valor mostrado é da saída da ULA e quando desativado, o valor mostrado é da saída do Program Counter;
- **KEY0:** Nesse caso a KEY0 serve como CLOCK do projeto, dessa forma ela faz com que a FPGA siga as instruções da ROM linha a linha de acordo com o clique no botão.