UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

INSTITUTO DE INFORMÁTICA

CURSOS DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO E ENGENHARIA DE COMPUTAÇÃO

EMANUEL NOVAKOSKI

GRÉGORI BARROS

THAYNÁ MINUZZO

VÍTOR TREVISAN

YURI JASCHEK

MIPS MONOCICLO, MULTICICLO E PIPELINE

Relatório referente ao Trabalho 1 da Disciplina de Organização de Computadores B.

Professor: Antônio Carlos S. Beck Filho

Porto Alegre

2018

INTRODUÇÃO

Este relatório visa resumir e explicar as alterações feitas nas três versões do MIPS: Monociclo, Multiciclo e Pipeline. Como o nosso grupo possui 5 integrantes, fizemos as 5 instruções selecionadas para o nosso grupo: JR, BNE, LUI, SRA e SLTI. Abaixo, um resumo das instruções e nas seções seguintes, explicações sobre o processo de implementação para cada instrução.

JR (Jump Register)
Formato: JR rs
Descrição: Programa incondicionalmente pula para o endereço no registrador rs.
Codificação dos bits:
31-26: SPECIAL (000000)
25-21: rs (5 bits)
20-6: 0 (15 bits)
5-0: JR (001000)

BNE (Branch On Not Equal)
Formato: BNE rs, rt, offset
Descrição: Se os valores de rs e rt forem diferentes, programa pula offset instruções, ou seja, PC é somado com offset shiftado com sinal 2 bits à esquerda.
Codificação dos bits:
31-26: BNE (000101)
25-21: rs (5 bits)
20-16: rt (5 bits)
15-0: offset (16 bits)

LUI (Load Upper Immediate)
Formato: LUI rt, immediate
Descrição: rt recebe immediate shiftado 16 bits à esquerda, concatenado com 16 bits 0.
Codificação dos bits:
31-26: LUI (001111)
25-21: 0 (5 bits)
20-16: rt (5 bits)
15-0: immediate (16 bits)

SRA (Shift Right Arithmetic)
Formato: SRA rd, rt, sa
Descrição: rd recebe o conteúdo de rt shiftado com sinal sa bits à direita.
Codificação dos bits:
31-26: SPECIAL (000000)
25-21: 0 (5 bits)
20-16: rt (5 bits)
15-11: rd (5 bits)
10-6: sa (5 bits)
5-0: SRA (000011)

SLTI (Set On Less Than Immediate)
Formato: SLTI rt, rs, immediate
Descrição: rt recebe 1 se o conteúdo de rs é menor que immediate estendido com sinal. rt recebe 0, caso contrário. A comparação é feita com sinal, através de subtração.
Codificação dos bits:
31-26: SLTI (001010)
25-21: rs (5 bits)
20-16: rt (5 bits)
15-0: immediate (16 bits)

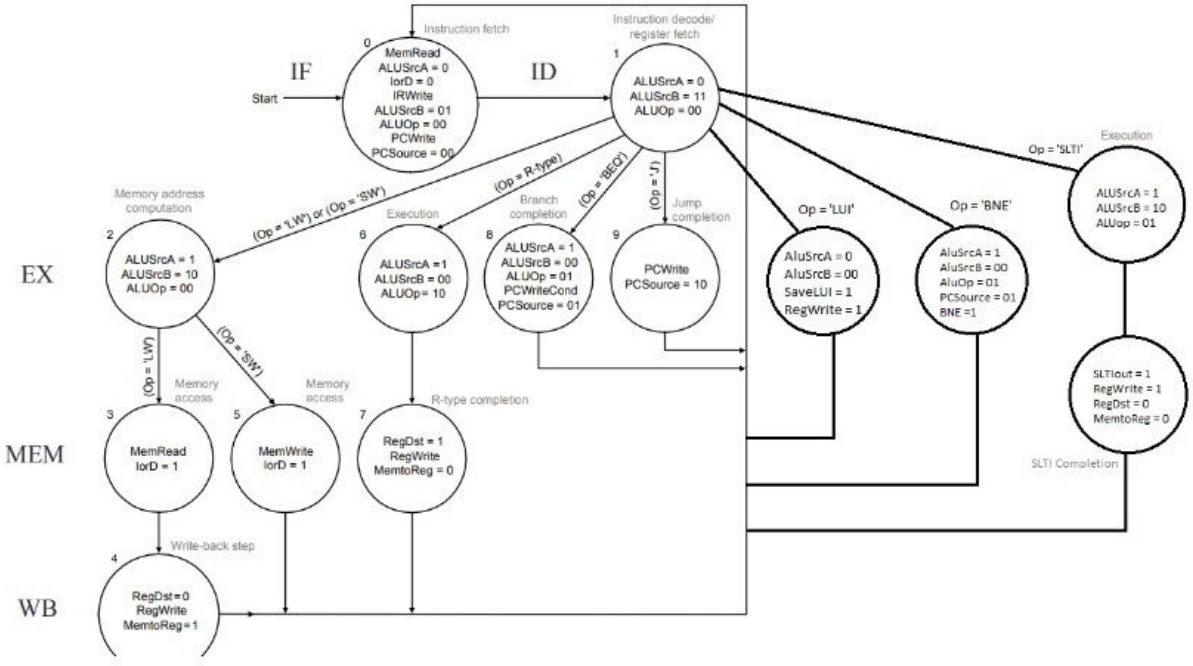
OVERVIEW

Tabela com os sinais de controle das versões monociclo e pipeline

Sinal\Instr	Tipo R	LW	SW	BEQ	J	BNE	LUI	SLTI
RegDst	1	0	X	X	X	X	0	0
Jump	0	0	0	0	1	0	0	0
Branch	0	0	0	1	X	1	0	0
MemRead	0	1	0	0	0	0	0	0
MemToReg	0	1	X	X	X	X	0	0
ALUop	10	00	00	01	XX	01	XX	01
MemWrite	0	0	1	0	0	0	0	0
ALUSrc	0	1	1	0	X	0	X	0
RegWrite	1	1	0	0	0	0	1	1
SaveLUI	0	0	0	0	0	0	1	0
BNE	0	0	0	0	X	1	0	0
STLI	0	0	0	0	0	0	0	1

As instruções JR e SRA estão contidas no Tipo R (SPECIAL opcode 000000)

Esquemático com os estados de controle da versão multiciclo



JR (Jump Register)

- Sempre que houver esta instrução, o PC deverá receber o valor lido do registrador rs. Foi criado, no monociclo, um multiplexador entre o PC e o multiplexador controlado pelo sinal Jump, já existente. O multiplexador recebe como entrada 0 a saída do multiplexador Jump e como entrada 1 o Read Data 1 do banco de registradores (que possui o conteúdo de rs).
- Como a instrução JR possui Opcode 000000, igual a outras do tipo R (como ADD, SUB etc), criamos o sinal que controla o multiplexador criado acima como uma saída do ALU Control. Este sinal só será 1 quando o controle mandar a ALU fazer o campo funct e o campo funct for JR (001000). Dessa forma, o PC será alterado para o conteúdo do registrador rs.
- Como JR tem 00000 no campo do rd e escritas no registrador $0 não alteram o estado do processador, a função funciona e está compatível com os sinais de controle das demais instruções do tipo R.
- Na versão multiciclo, o enable do PC é feito OR com o sinal da ALU Control, já que não é sempre ligado em 1.

BNE (Branch On Not Equal)

- Usando o hardware já existente para o BEQ, foi necessária apenas a criação de um sinal de controle, que chamamos de BNE. Foi feito AND deste sinal com a negação da saída zero da ALU. E OR desta AND com a AND do BEQ. Desta maneira, será feito branch quando for BEQ e o resultado da ALU for 0, ou quando for BNE e o resultado da ALU for não-zero.
- Os sinais de controle são os mesmos do BEQ, com exceção de que Branch=0 e BNE=1. Jump=X e todas as demais instruções devem ter BNE=0.

LUI

- Foi criado um hardware que shifta o immediate 16 bits à esquerda, concatenando com 16 bits de 0. Foi criado um multiplexador que recebe o valor do LUI na entrada 1 e o valor do multiplexador MemToReg na entrada 0. Foi criado o sinal de controle SaveLUI, que deve ser 1 na LUI e 0 nas demais. RegWrite=1 nesta instrução.

SRA (Shift Right Arithmetic)

- Como o opcode de SRA é 000000 (Tipo R), para decidir se o que será gravado no banco de registradores é a saída de MemToReg ou o novo hardware criado, que shifta com sinal o rt em sham unidades, foi criado um sinal de controle que é setado apenas se funct for SRA. Este novo multiplexador é usado, também, após MemToReg, pós ALU.

SLTI

- Após mandar a ALU fazer a subtração de rs com immediate estendido com sinal, o bit mais significativo do resultado será 1 se a diferença deu negativo, ou seja, se rs é menor que o immediate estendido com sinal, e 0 caso contrário.
- Este bit mais significativo, do resultado da ULA, é concatenado com 31 bits de 0s à esquerda (mais significativos). Esta é outra instrução onde deve ser adicionado um multiplexador antes da entrada de dados a serem escritos no banco de registradores. Para tal, foi criado o sinal SLTI.