JOHAN OSPINA
MICHAEL CARL ETHIER

Instruction fetch

Instruction decode/
register fetch

0

MemRead
ALUSrcA = 0
IorD = 0
IRWrite
ALUSrcB = 01
ALUOp = 00
PCWrite
PCSource = 00

Start

1

ALUSrcA = 0
ALUSrcB = 11
ALUOp = 00

Op = I-type

(Op = 'LW') or (Op = 'SW')

(Op = R-type)

(Op = 'BEQ')

(Op = 'J')

ALUSrcA = 1
ALUSrcB = 10
ALUOp = 00

Memory address
computation

Execution

Branch
completion

Jump
completion

2

ALUSrcA = 1
ALUSrcB = 10
ALUOp = 00

6

ALUSrcA = 1
ALUSrcB = 00
ALUOp = 10

8

ALUSrcA = 1
ALUSrcB = 00
ALUOp = 01
PCWriteCond
PCSource = 01

9

PCWrite
PCSource = 10

RegDst = 0
RegWrk = 1
MemtoReg = 0

(Op = 'LW')

(Op = 'SW')

Memory
access

Memory
access

R-type completion

ADD1 $S9, $S1, 9

3

MemRead
IorD = 1

5

MemWrite
IorD = 1

7

RegDst = 1
RegWrite
MemtoReg = 0

Write-back step

4

RegDst = 0
RegWrite
MemtoReg = 1

We have 3 main types of instructions. R-type, I-type, and J-type instructions. These use the hardware a bit differently to get the result we want. We are using two different memory modules. One is to store our instructions and then the other is to store the values of our instructions for later use, that way we don't have to worry about memory management and stack control.

R-type instructions == [31:26] Opcode | [25:21] R1 | [20:16] R2 | [15:11] R3
I-type instructions   == [31:26] Opcode | [25:21] R1 | [20:16] R2 | [15:0] Immediate
J-type instructions   == [31:26] Opcode | [25:0] jump address

JOHAN OSPINA
MICHAEL CARL ETHIER

Multicycle CPU — Jebediah Bush