Systems Architecture I

Exceptions

*This lecture was derived from material in the text (sec. 5.6).


Notes Courtesy of Jeremy R. Johnson
Introduction

• Objective: Insert support for exceptions into the multicycle implementation

• What’s New:
  – Need to detect exception
  – Need to transfer control to system to perform exception processing
  – Need to determine exception type
  – Need to resume control where exception occurred

• Topics
  – Exceptions (undefined instruction, arithmetic overflow)
  – Modifications to datapath
  – Modifications to control
Multicycle Datapath with Exception Support
Multicycle Control with Exceptions

-start

Instruction fetch

Instruction decode/ Register fetch

ALUSrcA = 0
ALUSrcB = 11
ALUOp = 00
PCSource = 00

ALUSrcA = 0
ALUSrcB = 00
ALUOp = 00
PCSource = 00

ALUSrcA = 1
ALUSrcB = 00
ALUOp = 10

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

PCWrite
PCSource = 10

MemRead
lOrD = 1

MemWrite
lOrD = 1

RegWrite
MemToReg = 1
RegDat = 0

Overflow

Write-back step

Memory address computation

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

(Op = R-type)

Branch completion

Jump completion

R-type completion

IntCause = 1
CauseWrite
ALUSrcA = 00
ALUSrcB = 01
ALUOp = 01
EPCWrite
PCWrite
PCSource = 11

IntCause = 0
CauseWrite
ALUSrcA = 00
ALUSrcB = 01
ALUOp = 01
EPCWrite
PCWrite
PCSource = 11

Memory access

(Op = 'BEQ')

(Op = 'J')

(Op = other)

Memory access

(Op = 'SW')

(Op = 'LW')

Memory access

(Op = 'BEQ')

(Op = 'J')

(Op = other)

Memory access

(Op = 'SW')

(Op = 'LW')

Memory access

(Op = 'BEQ')

(Op = 'J')

(Op = other)

Memory access

(Op = 'SW')

(Op = 'LW')

Memory access

(Op = 'BEQ')

(Op = 'J')

(Op = other)

Memory access

(Op = 'SW')

(Op = 'LW')

Memory access

(Op = 'BEQ')

(Op = 'J')

(Op = other)

Memory access

(Op = 'SW')

(Op = 'LW')

Memory access

(Op = 'BEQ')

(Op = 'J')

(Op = other)