Linh Nguyen

B04

ENCM 369 – Computer Organization

Lab 7

Ex. A) Part I

Using Figure 7.11 and Tables 7.1, 7.2, and 7.3 from the textbook, the control signals are:

Signal Value Meaning

MemtoReg 1 For lw, RD output of D-mem goes to R-file

MemWrite 0 Does not update D-mem

Branch 0 Not a branch instruction

ALUControl 010 ALU should add a GPR and an offset

ALUSrc 1 SrcB input to ALU is a sign-extended offset

RegDst 0 Bits 20:16 of lw specify destination GPR

RegWrite 1 lw writes to a GPR

The inputs A1, A2, and A3 are

A1 = 11101 ($sp)

A2 = A3 = 11001 ($t8)

The ALU inputs and the 32-bit result is

SrcA = 0x7fff\_fe90 (value of $sp)

SrcB = 0x0000\_0028 (sign-extended offset)

ALUResult = 0x7fff\_feb8 (sum of SrcA and SrcB)

WD3 input to R-file is

0x0003\_4567 (data at address 0x7fff\_feb8 from D-mem)

PCBranch Value is:

0x0040\_0154 + 4 + 40\*4 = 0x0040\_0154 + 4 + 0xa0

= 0x0040\_01f8

Part II

Signal Value Meaning

MemtoReg 0 For AND, ALUResult goes to R-file

MemWrite 0 Does not update D-mem

Branch 0 Not a branch instruction

ALUControl 000 ALU should do bitwise AND

ALUSrc 0 SrcB input to ALU is RD2 from R-file

RegDst 1 Bits 15:11 of AND specify destination GPR

RegWrite 1 AND writes to a GPR

The inputs A1, A2, and A3 are

A1 = 10010 ($s2)

A2 = 01011 ($t3)

A3 = 10010 ($s2)

The ALU inputs and the 32-bit result is

SrcA = 0x1002\_d364 (value of $s2)

SrcB = 0xffff\_ffc0 (value of $t3)

ALUResult = 0x1000\_d340 (bitwise AND of SrcA and SrcB)

WD3 input to R-file is the ALUResult: 0x1000\_d340

From Figure 7.11 of the textbook, PCBranch Value is given by the instruction address + 4 (0x0040\_015c) and the SignImm left-shifted by 2 bits as the input to an adder.

10010\_00000\_100100 (Bits 15-0 of instruction)

SignImm = 1111\_1111\_1111\_1111\_1001\_0000\_0010\_0100

SignImm << 2 = 1111\_1111\_1111\_1110\_0100\_0000\_1001\_0000

PCBranch:

11

0xfffe\_4090

+ 0x0040\_015c

-----------

0x003e\_41ec = PCBranch

Ex B)

Modifications to Main Decoder:

Output bits need to be added so that ALUOp is three bits wide instead of two and so that there is a SZ output bit that goes to the SZ input of the Extend unit.

Modifications to the ALU Decoder:

ALUOp input needs to be widened from two bits to three bits.

ALUOp is as follows:

Opcode ALUOp Meaning

R-type 010 Use funct field to decide ALU action

lw 000 add

sw 000 add

beq 001 subtract

addi 000 add

slti 100 set-less-than

andi 101 bitwise AND

ori 110 bitwise OR

Modified Table 7.2

ALUOp Funct ALUControl

000 X 010 (add)

001 X 110 (subtract)

010 100000 010 (add)

010 100010 110 (subtract)

010 100100 000 (and)

010 100101 010 (or)

010 101010 010 (set-less-than)

100 X 010 (set-less-than)

101 X 000 (and)

110 X 001 (or)

To create a table like table 7.3, the ALUOp values are taken from the previous work. SZ values are determined based on the facts that R-type instructions don’t use the output of Extend; lw, sw, beq, and addi use the 32-bit sign-extension of bits 15-0 of the instruction; andi and ori use the 32-bit zero-extension of bits 15-0 of the instruction.

Modified Table 7.3

Instr Opcode RegWrite RegDst ALUSrc Branch MemWrite MemtoReg ALUOp SZ

R-type 000000 1 1 0 0 0 0 010 X

lw 100010 1 0 1 0 0 1 000 1

sw 100011 0 X 1 0 1 X 000 1

beq 000100 0 X 0 1 0 X 001 1

addi 001000 1 0 1 0 0 0 000 1

slti 001010 1 0 1 0 0 0 100 1

andi 001100 1 0 1 0 0 0 101 0

ori 001101 1 0 1 0 0 0 110 0

Ex C)



