ECE 3055

Midterm I- Solution Key
June 13th, 2006

1. Consider the execution of the following block of SPIM code. The text segment starts at 0x00400000 and that the data segment starts at 0x10010000.

```
        .data
start:  .word 0x32, 33, 0x34, 35
end:    .space 16
str:    .asciiz "Test 2"
        .align 3
mask:   .byte 0xFF, 0x00, 0x00, 0x00
        .text
        .globl main
main:   li $t3, 4
        la $t0, start
        la $t1, end
        la $t2, mask
        lw $t3, 0($t2)
loop:   lw $t4, 0($t0)
        and $t4, $t4, $t3
        sw $t4, 0($t1)
        addi $t0, $t0, 4
        addi  $t1, $t1, 4
        addi $t3, $t3, -1
        bne $t3, $zero, loop
exit:   li $v0, 10
        syscall
```

   a. Show the corresponding contents of the locations in the data segment **given below**. **Note the specific addresses!** Assume memory is initialized to 0x00000000 before the code is loaded.

Address	Contents
0x10010000	0x00000032
0x10010004	0x00000021
0x10010008	0x00000034
0x1001000c	0x00000023
0x10010020	0x74736554
0x10010024	0x00003220
0x10010028	0x000000FF
0x1001002c	0x00000000

   b. In a big Endian machine, what will be contents of $t4 the first time through the loop, i.e., the first time the branch instruction is encountered?

___0x00000000_____

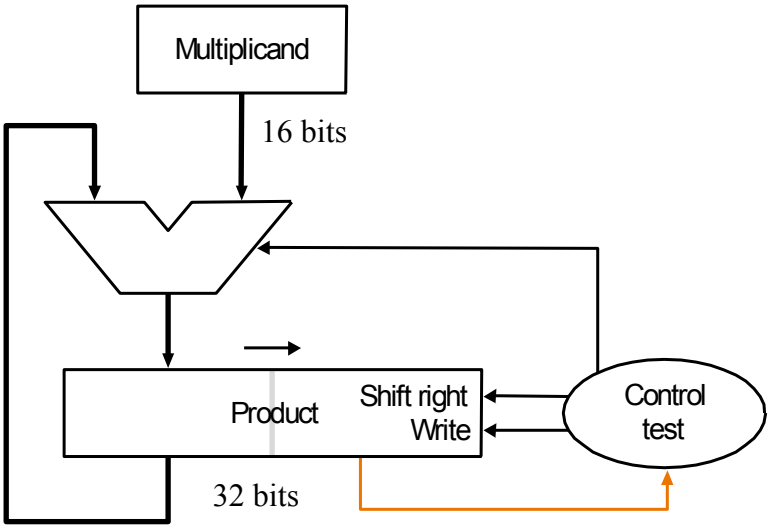
   c. What is the value of exit? Your answer must account for how pseudo instructions are translated.

__0x 00400038_____

2. The following is the binary representation of a block of assembled SPIM code. Disassemble the program producing the original SPIM program. The opcode map on the bottom of page 12 is the most helpful.

Assembled Binary	SPIM Instruction
0x8f880000	lw $8, 0($28)
0x01094820	add $9, $8, $9
0x239c0004	addi $28, $28, 4
0x1500fffc	bne $8, $0, FFFC

3. The following question deals with the unsigned multiplier below from chapter 3.



a. Assume you are performing the following operation: 0x00ac * 0x1010 (the former is the multiplicand). What is the contents of the product register after the completion of the 6th clock cycle.

_____0x002b0040_____

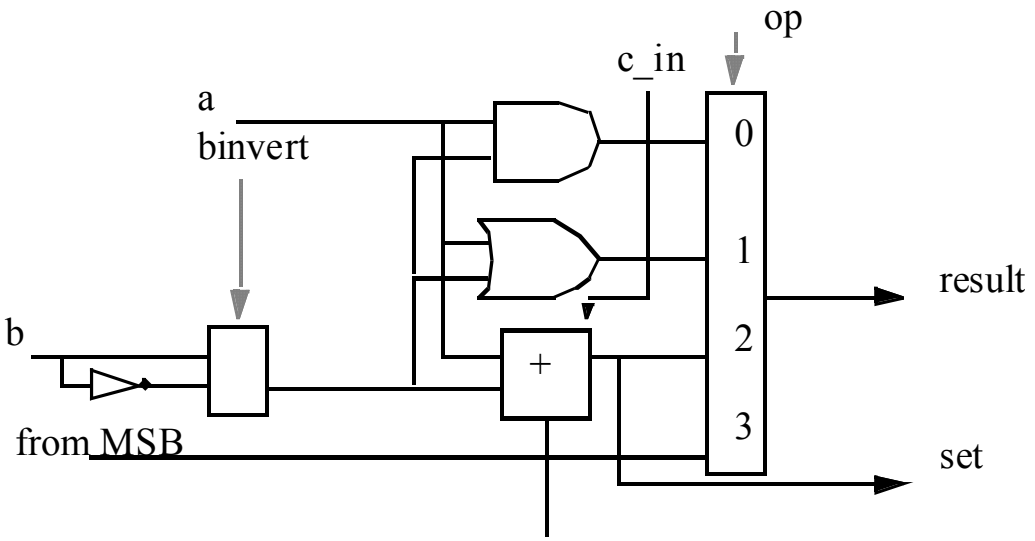
b. Consider an ALU constructed as shown in class (bit position 0 is shown below). Assume that a ripple carry implementation where each bit position introduces 2 gate delays in the carry chain and the output multiplexor introduces 2 gate delays. What is the delay of a `slt` instruction (ignore the cost of inverting b)?

In a slt operation, the two operands are subtracted and the msb of the operation is used as the lsb of the result.

Delay is calculated as:

2 gate delay for each bit in the adder chain to propagate the carry
= 2 * 32
+
2 gate delay for last mux (other mux delays are overlapped/parallel)

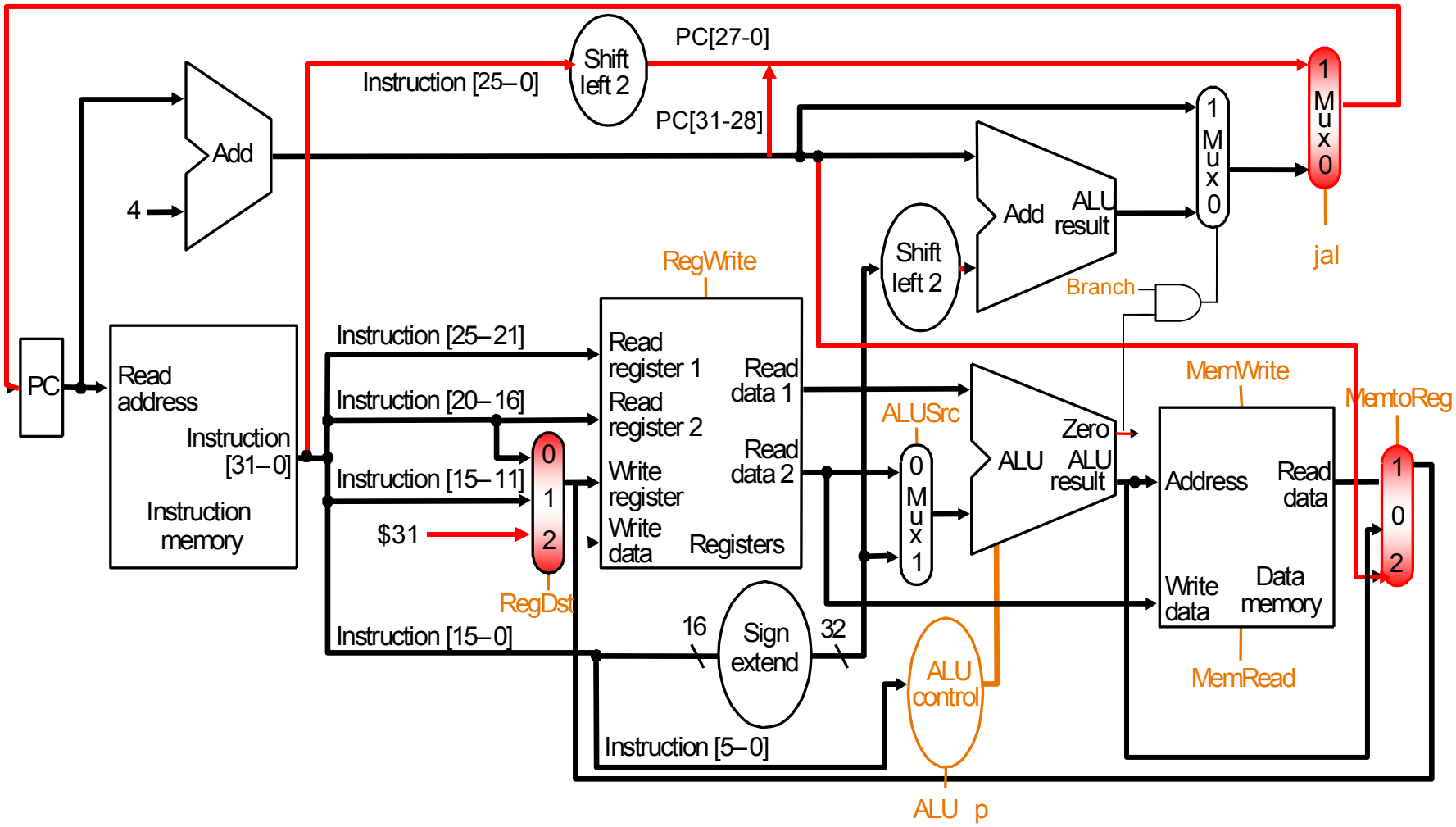
= 64 + 2 = 66 gate delays



4. Consider the single cycle datapath shown overleaf. We wish to modify this datapath to include the jal instruction. Answer the following with respect to this goal.
    a. Modify the datapath (clearly markup the figure) to show how the jal instruction can be implemented. Add additional signals/components and modify existing ones as necessary.
    b. Show the values of all of the control signals that will implement the jal instruction after your modifications. If you have added new control signals, specify their values. You can enter these values in the table below the datapath figure.

The additions are shown in the following figure.

As explained in class, to extend the datapath for a jal instruction:
- a new control signal (jal) will be added to the control unit.
- a new mux is added to the path that will control whether the PC is updated with the jump address - 26 bit word address from the instruction, left shift by 2 (*4) to get the byte address, and upper four bits set to that of the PC.
- The mux that controls the write register address must be extended to include the hard coded value of 31 (the return address register). The control signal for this mux is increased by 1 bit.
- The result of PC+4 must be input to the mux that selects the data to write to the register. The control signal for this mux is increased by 1 bit.



Instruction	RegDst	ALUSrc	MemToReg	RegWrite	MemRead	MemWrite	Branch	ALUOp	Jal			
Jal	10	X	10	1	X	0	X	XX	1			

5. Consider the multi-cycle datapath shown overleaf executing the code shown below. Assume that the la, li, and addi instructions each take 4 cycles,

```
        .data
start:  .word 0x32, 33, 0x34, 35
end:    .space 16
str:    .asciiz "Test 2"
        .align 3
mask:   .byte 0xFF, 0x00, 0x00, 0x00
        .text
        .globl main
main:   li $t3, 4
        la $t0, start
        la $t1, end
        la $t2, mask
        lw $t3, 0($t2)
loop:   lw $t4, 0($t0)
        and $t4, $t4, $t3
        sw $t4, 0($t1)
        addi $t0, $t0, 4
        addi  $t1, $t1, 4
        addi $t3, $t3, -1
        bne $t3, $zero, loop
exit:   li $v0, 10
        syscall
```

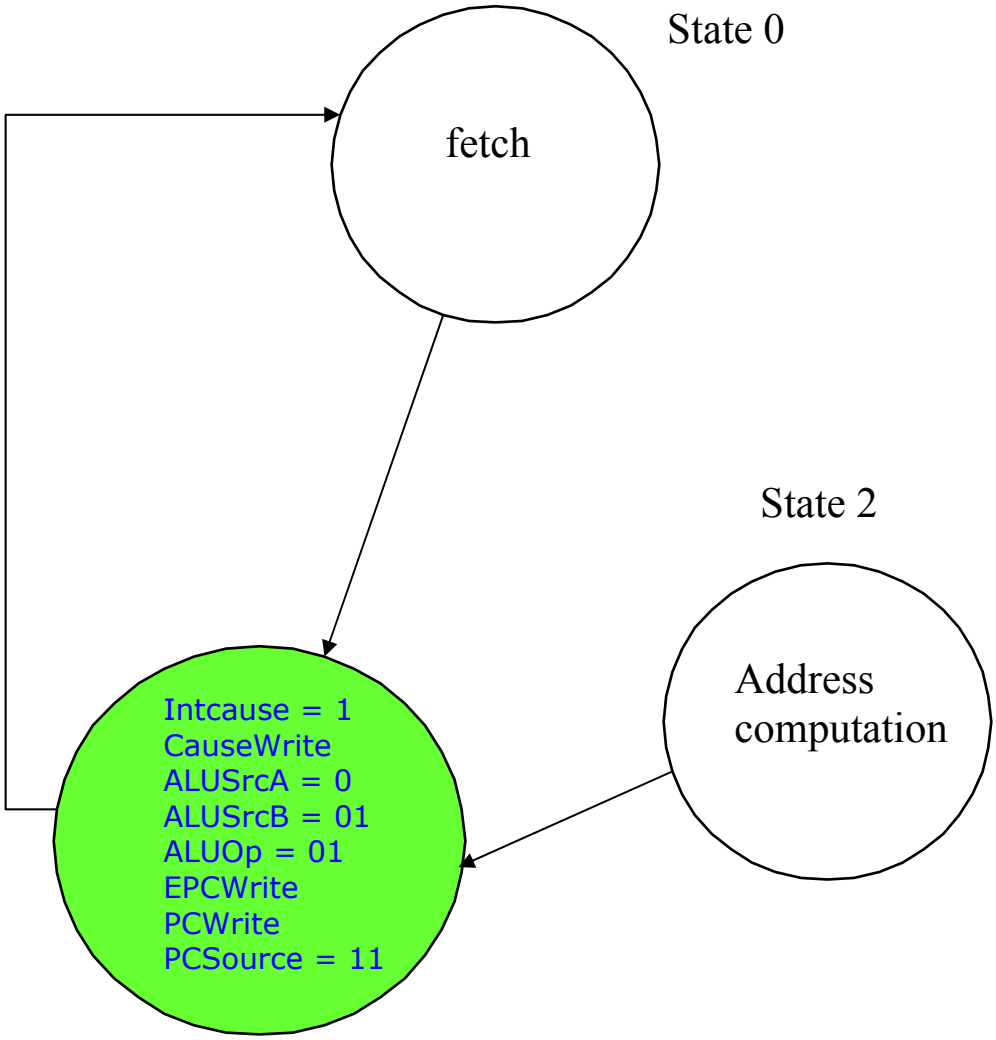
a. Fill in the values of the following signals on cycle 48. **The first cycle is cycle 0!**

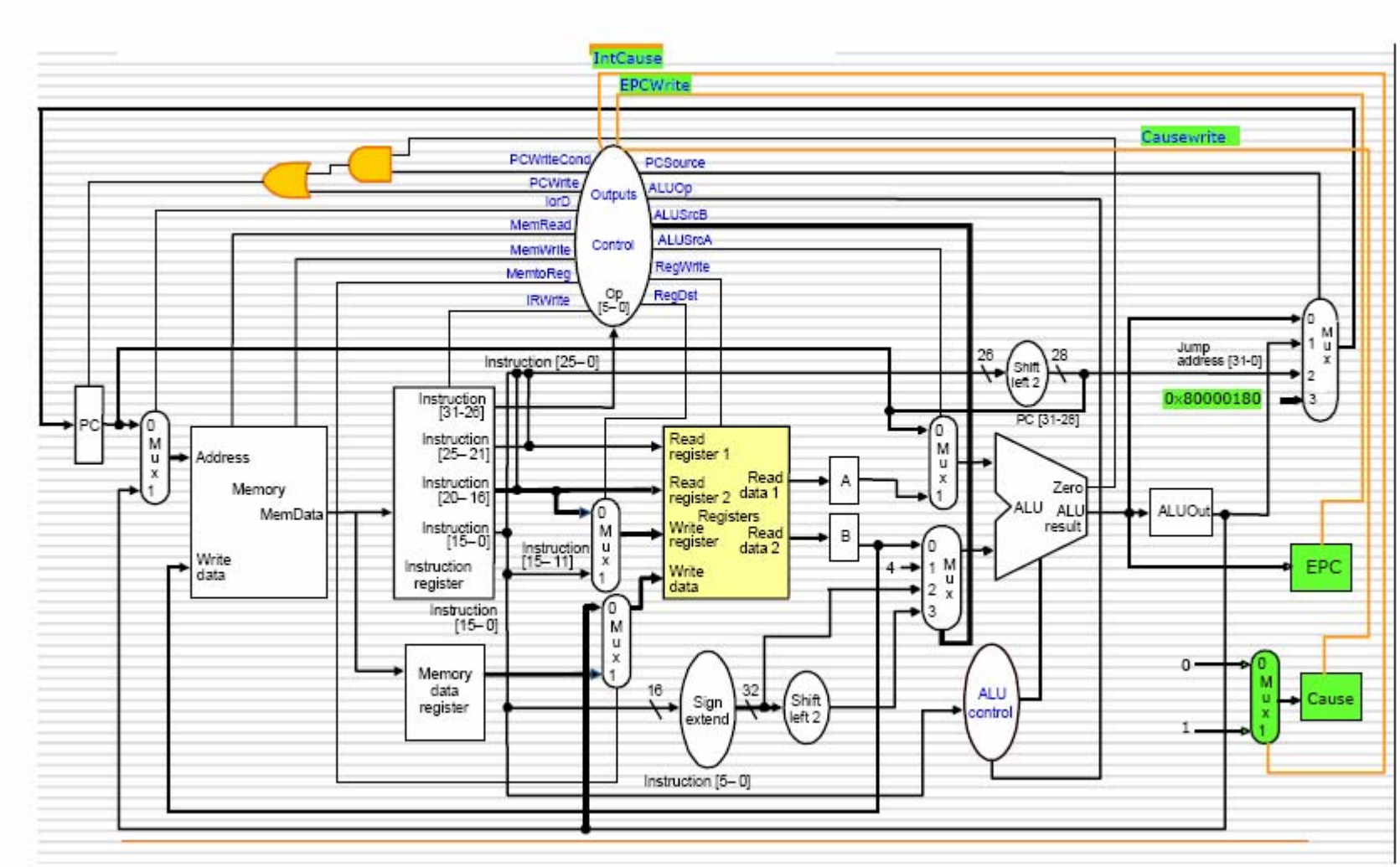
	PCWrite	ALUOp	RegDst	MemToReg	Zero	ALUOut (contents)
Cycle 48	0	01	x	x	0	0x0040001c

Cycle number 48 is the execution cycle of the branch instruction.

b. Consider a memory address exception where the area of memory being addressed is on disk rather than in memory. Thus we need the OS to intervene to move the page to memory and restart the execution of the instruction. Modify the datapath overleaf to i) transfer control to an exception handler at address 0x08000010, and ii) store the return address (the address of the offending instruction) in an exception program counter, and ii) store the cause (cause = 1) in a Cause register. Below draw a modified state machine for the multi-cycle datapath showing the additional states necessary for exception processing. You only define the control signal values for the exception states.

Memory address exeptions may be encountered on instruction fetch or address computation for a load/store. In this figure Cause =1 corresponds to an address exception. The Program Counter stores the address of the offending instruction.





For Exception Handler