



3. (15 points)
- a. (10 points) There are about 25 students in ECE 154.
    - i. (3 points) How many bits are required to represent 25154 in two's complement notation?
  
  
  
  
  
  
  
  
  
  
    - ii. (3 points) If every student is to be assigned a unique bit pattern, what is the minimum number of bits required to do this?
  
  
  
  
  
  
  
  
  
  
    - iii. (4 points) How many more students can be admitted to the class without requiring additional bits?

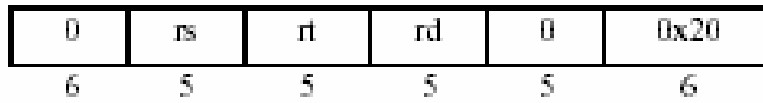
- b. (5 points) Suppose a 32-bit instruction takes the following format:

| opcode | SourceReg | DestinationReg | Immediate Value |

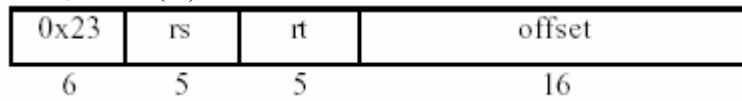
If there are 60 opcodes and 32 registers, what are the maximum and minimum values that the immediate (IMM) can take (assume Immediate Value is a 2's complement value)?



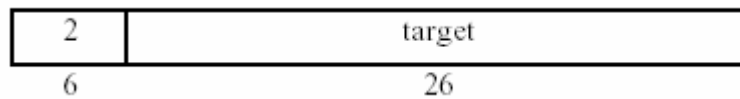
a. add rd, rs, rt



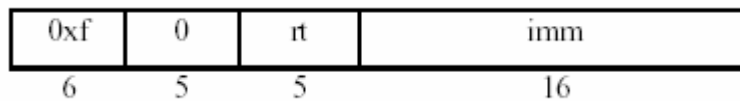
b. lw rd, offset(rs)



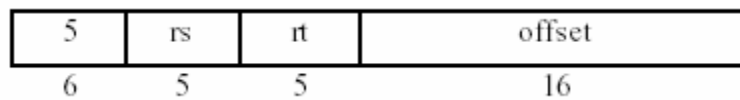
c. j target



d. lui rt, imm



e. bne rs, rt, label



5. (20 points) The original reason for Booth's algorithm was to reduce the number of operations by avoiding operations when there were string of 0s and 1s. Revise the algorithm presented in class to look at 3 bits at a time and compute the product two bits at a time.

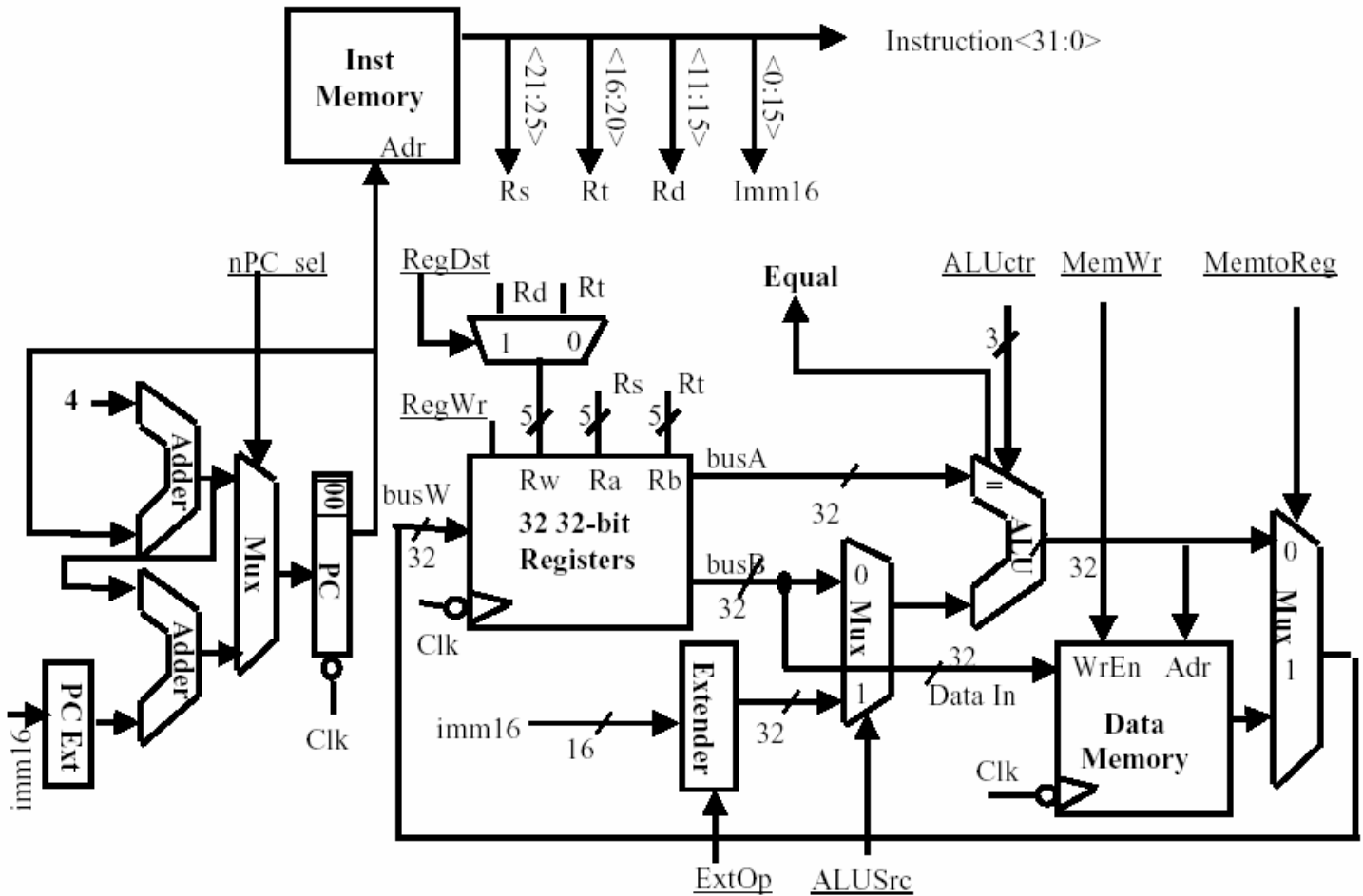
Recall the table for 2 bits (A is the multiplicand, B is the multiplier):

$A_i$	$A_{i-1}$	Operation
0	0	Do nothing
0	1	Add B
1	0	Subtract B
1	1	Do nothing

Fill in the following table to determine the 2-bit Booth encoding. Assume that you have both the multiplicand and 2 x the multiplicand already in registers. Explain the reason for the operation on each line. Two of the cases are given for you.

Current Bits		Previous Bit	Operation	Reason
$A_{i+1}$	$A_i$	$A_{i-1}$		
0	0	0	None	Middle of a string of 0s
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1	None	Middle of a string of 1s

6. (15 points)



a. (10 points) Fill in the following table for control signals for the following instructions for given datapath

**ADDU:**  $R[rd] \leftarrow R[rs] + R[rt]$ ;  $PC \leftarrow PC + 4$

**LOAD:**  $R[rt] \leftarrow MEM[R[rs] + \text{sign\_ext}(Imm16)]$ ;  $PC \leftarrow PC + 4$

	RegDst	nPCsel	ExtOp	MemtoReg	ALUOp	MemWr	ALUSrc	RegWr
ADDU								
LOAD								

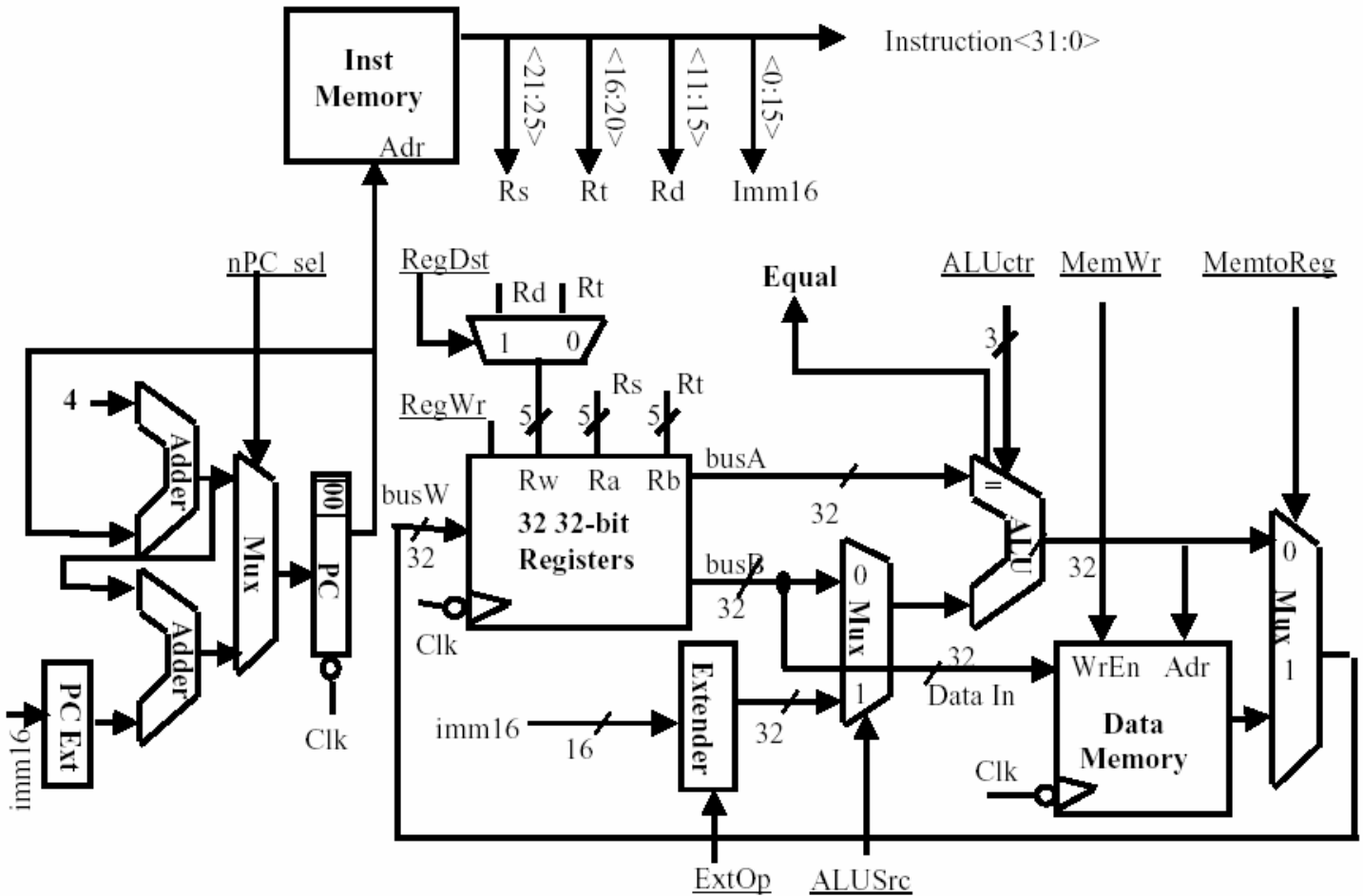
ALUOp can be add, subtract, or  
 MemWr and RegWr are high when writing

Name \_\_\_\_\_  
Student ID \_\_\_\_\_

Page 7

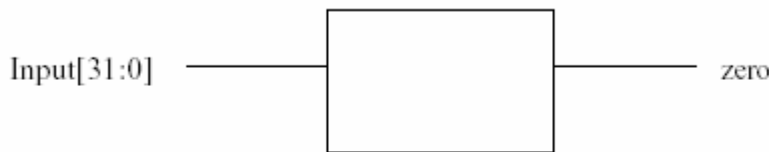
b. (5 points) Draw the critical path for addu on the datapath using a dashed line and describe it below (e.g. PC->Instr Mem->....)

7. (20 points)



Suppose we wish to add support for the new instruction, `bmez`: branch if memory equals zero, into the datapath shown above. The format of the instruction is as follows, `bmez $rt, $rs, imm16`

The instruction reads the value of memory in address  $(R[\$rs] + R[\$rt])$ , compares it to zero and if zero then branches to the zero extended immediate, otherwise it goes on to the next instruction. You can add logic gates and muxes, and the zero compare block below, and one new control signal, `bmCntrl`.



zero is high when the Input is zero.

