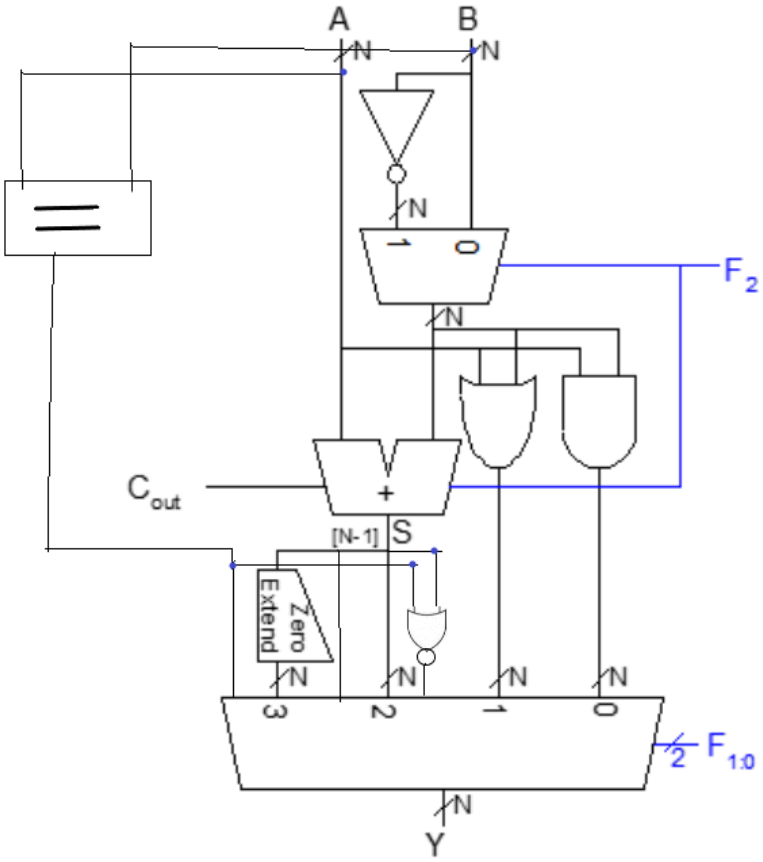
Name-Last Name: Şevval Atmaca **Student ID:** 21827115

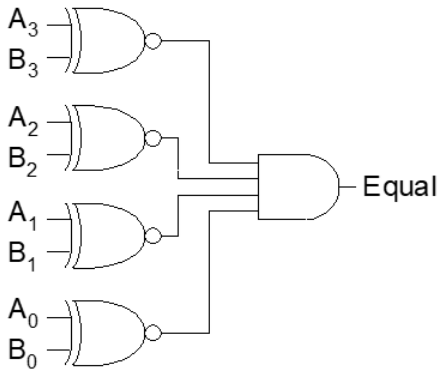
Hacettepe University	Department of Computer Engineering
BBM234 Computer Organization Spring 2021	Instructor: Prof. Dr. Suleyman TOSUN
Homework 1	
Assigned date: 28.03.2021	**Submission deadline: 05.04.2021 at 13:59:59 via submit.cs.hacettepe.edu.tr**

Q1. Consider the ALU given below. This ALU has N-bit A and B inputs and 3-bit F input to control the operation mode of the ALU. It outputs N-bit output Y based on the selected operation. We would like to add more functionality to this ALU.

[25 pts] Add three 1-bit outputs to the ALU: *LT* (it is 1 when A is less than B), *EQ* (it is 1 when A is equal to B), and *GT* (it is 1 when A is greater than B).
Show your additional circuits on the ALU and explain them for full credit.

$F_{2:0}$	Function
000	A & B
001	A | B
010	A + B
011	not used
100	A & ~B
101	A | ~B
110	A - B
111	SLT





Implementation of equality comparator

LT: To decide A is less than, I use substraction. If a<b, then a – b <0. If the result of a-b is negative, when you convert it to binary, the negative binary number's most significant bit is 1. Accordingly, If the result is 1 then less than, otherwise not.

ET: To decide A is equal to B, I use equality comparator. If the two numbers are equal, then the nth bits of these two numbers are the same. This comparator compares all bits and if they are same the result is 1, otherwise 0.

GT: To decide A is greater than, I use substraction and equality comparator. if a> b, then a-b>0. Since, the result of a-b will be positive, the most significiant bit will be 0 as the binary number. But also 0 when the numbers are equal, and I used the equality comparator to distinguish it. If a number results in 0 from EQ (i.e. not equal) and the most significant bit of the result of a-b is 0, then a is greater than b.

Q2.

(a) [9 pts] For 8-bit signed integer x = 0xA2, do the following calculations. Write your results in decimal.

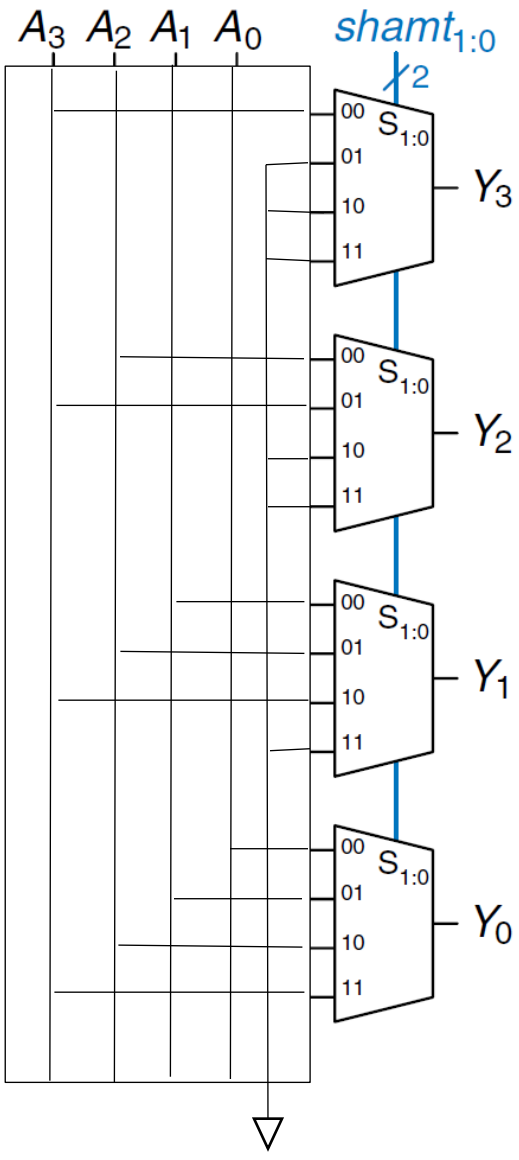
x >> 4 = 10

x << 4 = 32

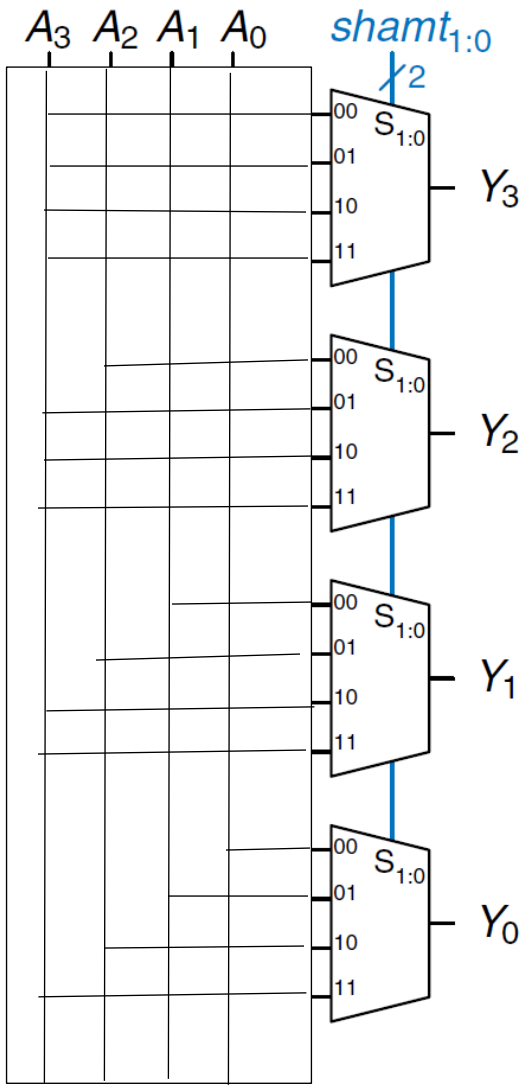
x >>>4= -6

(b) [16 pts] Below you will implement a logical and arithmetic shifter. The output *Y* will be the input *A* shifted by 0 to 3 bits depending on the value of the 2-bit shift amount $shamt_{1:0}$. For both shifters, when $shamt_{1:0}$= 00, *Y*=*A*.

Logical Right Shift



Arithmetic Right Shift



Q3. [25 pts] Write the machine code for the instructions given in bold. Indicate their instruction type, fill the binary machine code by showing the corresponding fields, and write their hexadecimal values into the boxes.

Address	Instruction	Opcode or funct	Register numbers
0x90 fact:	addi $sp, $sp, -8		
0x94	**sw $a0, 4($sp)**	#opcode: 0x2B	a0: 4, sp: 29
0x98	sw $ra, 0($sp)		
0x9C	addi $t0, $0, 2		
0xA0	slt $t0, $a0, $t0		
0xA4	**beq $t0, $0, else**	#opcode: 0x04	t0: 8
0xA8	addi $v0, $0, 1		
0xAC	addi $sp, $sp, 8		
0xB0	**jr $ra**	#funct: 0x08	ra: 31
0xB4 else:	addi $a0, $a0, -1		
0xB8	**jal fact**	#opcode: 0x03	
0xBC	lw $ra, 0($sp)		
0xC0	lw $a0, 4($sp)		
0xC4	addi $sp, $sp, 8		
0xC8	mul $v0, $a0, $v0		
0xCC	jr $ra		

Instruction	Type	Binary Code	Hex code
sw $a0, 4($sp)	I	101011 11101 00100 0000000000000100	0xAFA40004

Instruction	Type	Binary Code	Hex code
beq $t0, $0, else	I	000100 00000 01000 0000000010110100	0x100800B4

Instruction	Type	Binary Code	Hex code
jr $ra	R	000000 11111 00000 00000 00000 001000	0x03E00008

Instruction	Type	Binary Code	Hex code
jal fact	J	000011 00000000000000000010010000	0x0C000090

Q4. [25 pts] You have four instructions stored in the memory as given in the following table:

Instructions	Address	Instruction
Inst1	0x00400000	0x3308FFF8
Inst2	0x00400004	0x12000002
Inst3	0x00400008	0x01098020
Inst4	0x0040000C	0x08100001
Inst5	0x00400010	---

a) Write the binary values for each instruction. Clearly show which bits corresponds to which field in the instruction format (opcode, rs, rt, rd, etc.).

Instructions	Instruction format
0x3308FFF8	001100(op) 11000(rs) 01000(rt) 1111111111111000(imm)
0x12000002	000100(op) 10000(rs) 00000(rt) 0000000000000010(imm)
0x01098020	000000(op) 01000(rs) 01001(rt) 10000(rd) 00000(shamt) 100000(func)
0x08100001	000010(op) 00000100000000000000000001(addr)

b) Write down the corresponding MIPS assembly code below for each machine code.

Instructions	MIPS Code
Inst1	andi $t0 $t8 0xFFF8
Inst2	beq $0 $s0 0x0002
Inst3	add $s0 $t0 $t1
Inst4	j 0x0100001

Name	Register
$0	0
$at	1
$v0-$v1	2-3
$a0-$a3	4-7
$t0-$t7	8-15
$s0-$s7	16-23
$t8-$t9	24-25
$k0-$k1	26-27
$gp	28
$sp	29
$fp	30
$ra	31

Instruction	Opcode
j	000010
jal	000011
beq	000100
bne	000101
addi	001000
slti	001010
andi	001100
ori	001101
xori	001110
lui	001111
lw	100011
sw	101011

Instruction	Funct
sll	000000
srl	000010
sra	000011
jr	001000
div	011010
add	100000
sub	100010
and	100100
or	100101
xor	100110
nor	100111
slt	101011