Yellow

NAME _____
First Last

Student ID# _____

STONY BROOK UNIVERSITY

COMPUTER SCIENCE DEPARTMENT

MIDTERM EXAMINATION
VERSION A

CSE 320
Fall 2014
October 14, 2014

This is a closed-book exam. (80 minutes)
Use this form for your work and return it.

The exam has 8 problems.
It is crucial to show all work done on the provided paper.

#1 _____	(13 pts)	#6 _____	(10 pts)
#2 _____	(10 pts)	#7 _____	(8 pts)
#3 _____	(12 pts)	#8 _____	(12 pts)
#4 _____	(12 pts)	#9 _____	(15 pts)
#5 _____	(8 pts)		

TOTAL _____ (100 Max)

1 [13pts] Short Answer

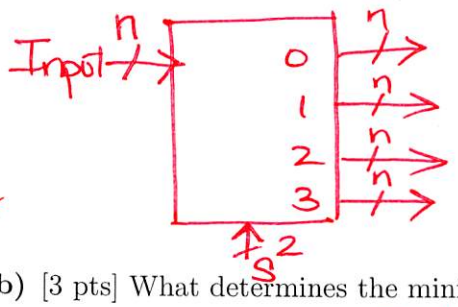
(a) [3 pts] What is the difference between a decoder and an encoder?

Decoder takes an n-bit binary # and places a 1 on the corresponding output line

-2 switched

An encoder, encodes the 1 on the inputs into the n-bit binary # of that line

(b) [3 pts] Draw and explain the operation of a 2-bit selector demultiplexor?



+1 mux modif +2 1 bit

the n-bits on Input are placed on the n-bit output line based on the selector bits

+1.5 diagram only +2 bad explanation

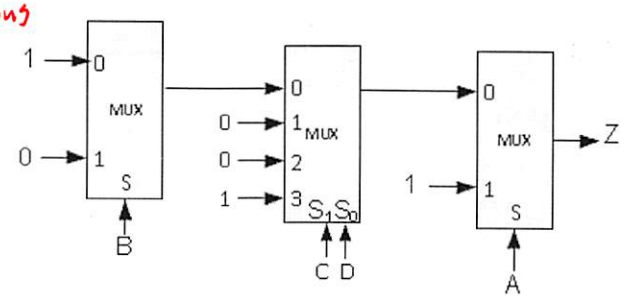
(b) [3 pts] What determines the minimum clock cycle time of a circuit?

The Instruction that takes the longest to Execute.

length of the critical path of the - boolean logic - circuit - combinatorial logic

(d) [4 pts] What is the Boolean equation for the following circuit? Do not minimize.

-1 small mistake (missing ')
-2 1 or more terms wrong
-3 incorrect terms, and instead of +

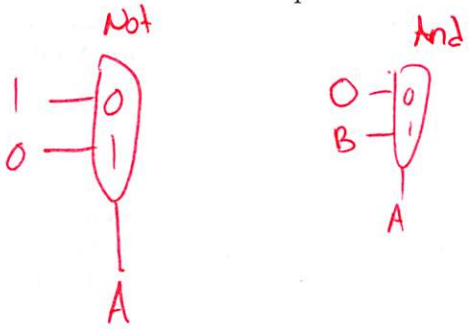


$A + A'(CD + B'C'D') = Z$

2 [10 pts] Short Answer

(a) [5 pts] Demonstrate how a 2-bit multiplexor is a universal gate.

-2 not universal
-2 2 bit selectors
-2 wrong imp. of gate

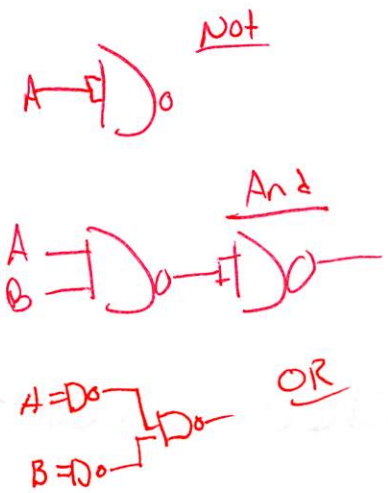


A	B	A'	AB
0	0	1	0
0	1	1	0
1	0	0	0
1	1	0	1

AB

A B	NAND	NOR
0 0	1	1
0 1	1	0
1 0	1	0
1 1	0	0

(b) [5 pts] Demonstrate how NAND is a universal gate.

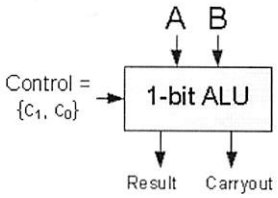


3 [12 pts] ALUs

(a) [4 pts] Using 1-bit ALUs we make an 8-bit ALU (operates on 8 bit numbers) and a 16-bit ALU (operates on 16-bit numbers). Consider A AND B, A + B (addition) and A OR B performed by these ALUs.
Fill in the blank with "less", "same", or "more"

(i) An 8-bit ALU requires ___less___ time for A + B as a 16-bit ALU.
(ii) A 16-bit ALU requires ___same___ time for A AND B than an 8-bit ALU.
(iii) An 8-bit ALU requires ___more___ time for A + B than a 16-bit ALU doing A AND B.
(iv) A 16-bit ALU requires ___same___ time for A AND B than an 8-bit ALU doing A OR B

(b) [8 pts] Using 2-bit (1-selector) multiplexors and 1-bit Adders, build a 2-bit ALU as described by the figure (with $Carry_{out}$ ~~and Zero output signals~~) which performs the following functions. Numbers are in two's complement form. Note: There is no $Carry_{in}$ signal to the larger unit.



Control	Operation
00	A + B
01	A AND B
10	A - B
11	B - A

- See attached Sheet -

Ver A

4 [12 pts] Boolean Expressions.

$$f = (AB')'(A + C)(A'BC)'$$

(a) [2 pts] How many literals appear in the boolean expression? 7

(b) [4 pts] Find all maxterms, answer in boolean expression form:

$(A'+B^{D})(A+C)(A+B'+C')$

$A\,A'\,A'\,C\,A$

$(A'+B+C)(A'+B+C')(A+B'+C)(A+B+C)(A+B'+C')$

4 5 2 0 3

-1 (missing or incorrect term)

2 Maxterms

(c) [2 pts] Name the minterms, answer in $\sum M(?)$ form : 1 & 0

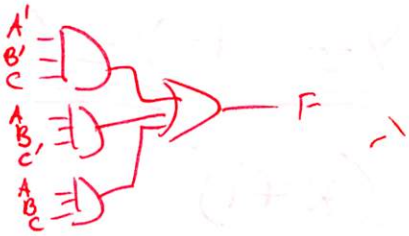
-.5 for missing or wrong term -1 for maxterms -.5 not $\sum$

$\sum m(1,6,7)$

001

(d) [2 pts] Draw the 2-level AND-OR or OR-AND gate network for part (c). Do not minimize the expression.

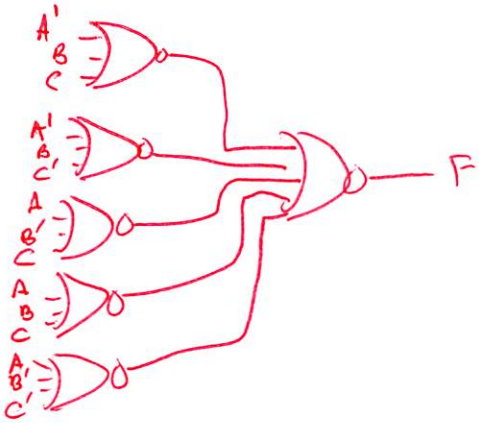
$A'B'C + ABC' + ABC$



-.5 / wrong 1st level gate -1

(e) [2 pts] Draw the 2-level NAND-NAND or NOR-NOR gate network for part (b). Do not minimize the expression.

-.5 for wrong 1st gate input



-1 inverse

0 -1/0
1 -2/1

5 [8 pts] Using the postulates of Boolean algebra, minimize the following:

$$X'Y' + Y'Z + XZ + XY + YZ' = X'Y' + XZ + YZ'$$

+1 did 1 valid thing

+2 made up all their rules

+1 comment follow

+5 made up 1 rule at end / or stopped

+3 did a reduction, then stopped

+3 some valid, but a large fudge

+7 unclear steps

$X'Y' + Y'Z(X+X') + XZ + XY + YZ'$

$X'Y' + XY'Z + X'Y'Z + XZ + XY + YZ'$

$X'Y'(1+Z) + XZ(Y'+1) + XY + YZ'$

$X'Y' + XZ + XY(Z+Z') + YZ'$

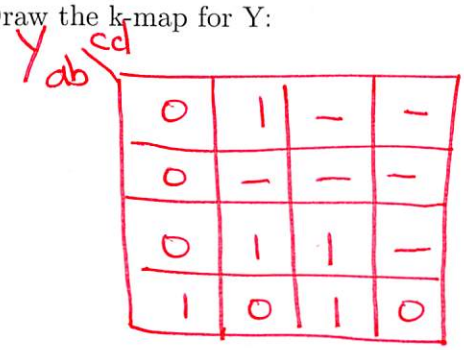
$X'Y' + XZ + XYZ + XYZ' + YZ'$

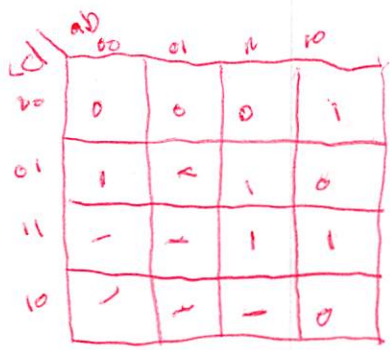
$X'Y' + XZ(1+Y) + YZ'(X+1)$

6 [10 pts] Consider the following truth table for Y:

a	b	c	d	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	-
0	0	1	1	-
0	1	0	0	0
0	1	0	1	-
0	1	1	0	-
0	1	1	1	-
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	-
1	1	1	1	1

(a) [2 pts] Draw the k-map for Y:



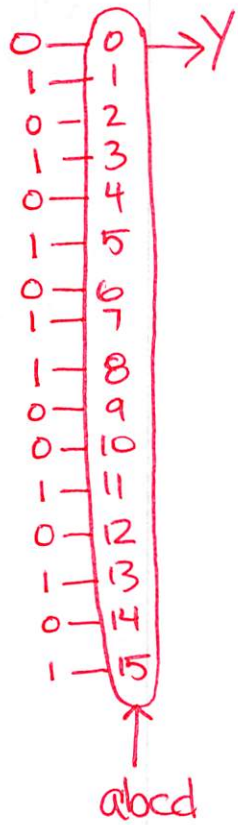


(b) [2 pts] Write the minimal SOP expression for Y:

$Y = ab'c'd' + a'd + bd + cd$

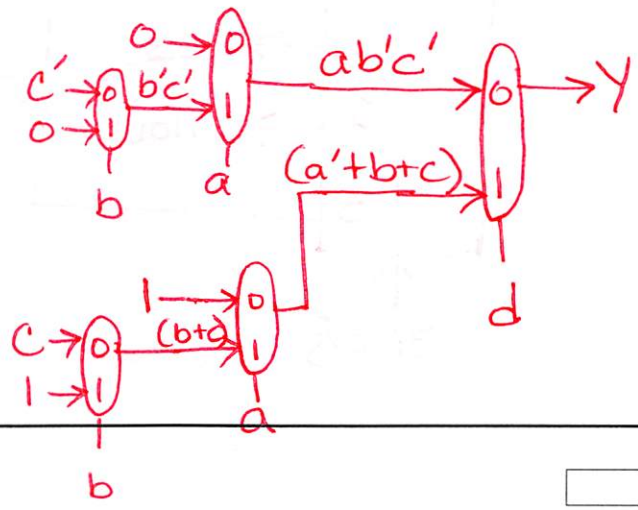
(c) [2 pts] Implement the expression from part (b) with a single 4-bit selector mutiplexor

Can not have DC in circuit! Based on (b), the DC became 1 or 0.

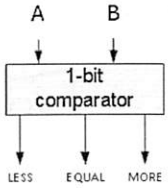


(d) [4 pts] Implement the expression from part (b) with a 2-bit mutiplexors

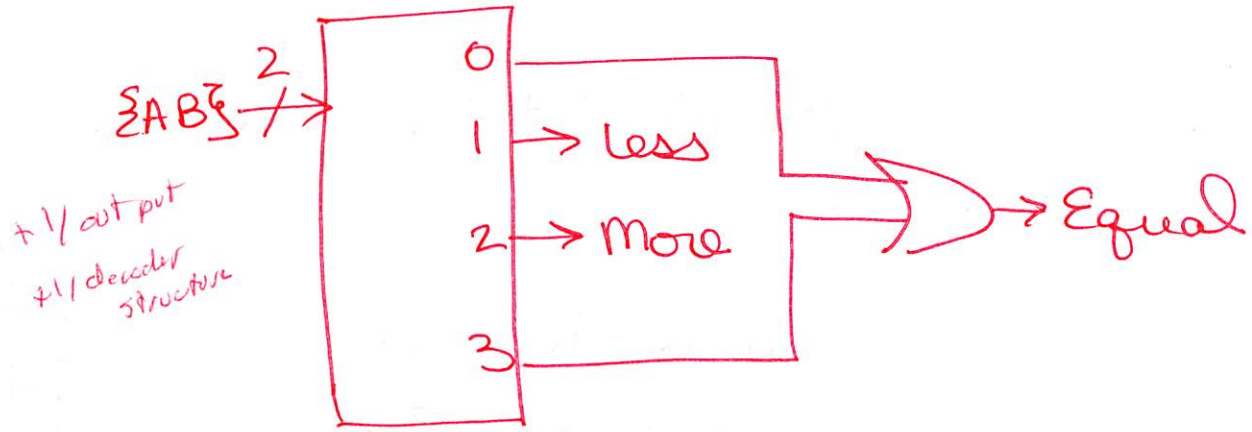
one possible solution



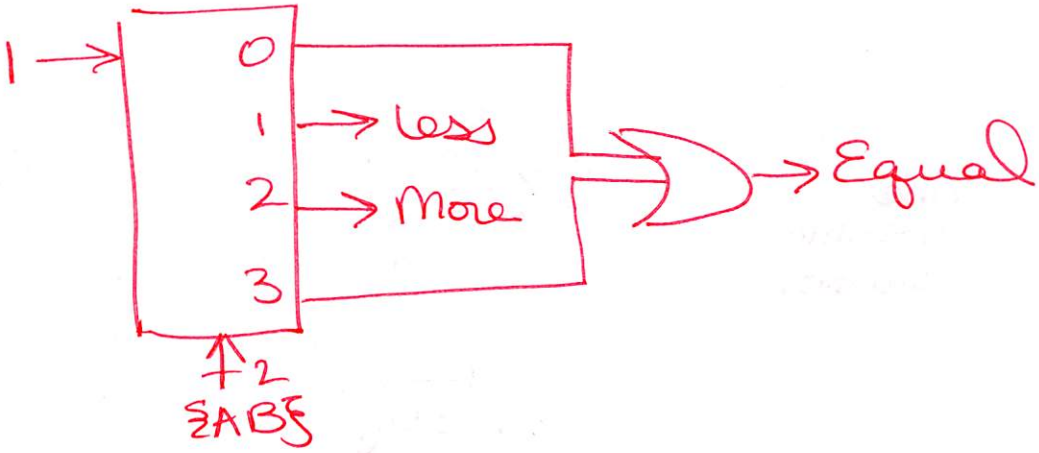
7 [8 pts] Consider the 1-bit comparator unit as specified below.
LESS = 1 when A < B, EQUAL = 1 when A == B, MORE = 1 when A > B



(a) [4 pts] Implement the unit using a 2-bit decoder and OR gates.



(b) [4 pts] Implement the unit using a 2-selector demultiplexor and OR gates.



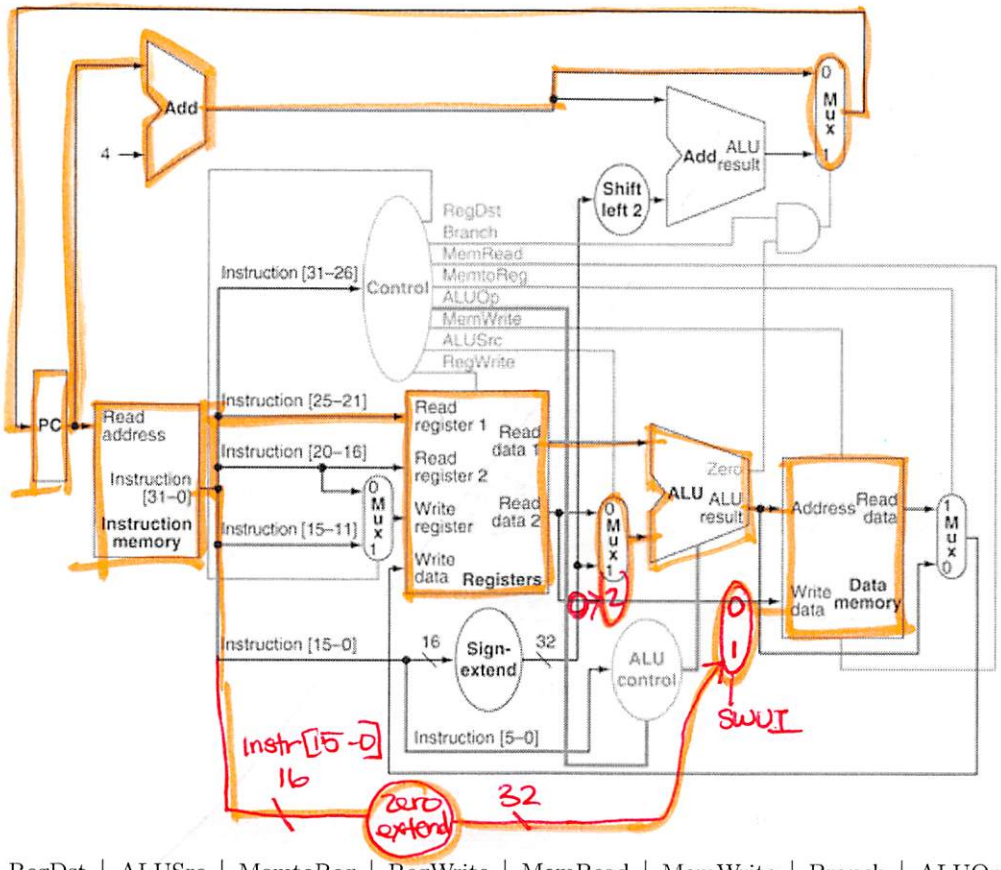
8 [12 pts] Single-cycle Datapath

Implement a "store word unsigned immediate" instruction into the single-cycle MIPS datapath.

```
SWUI Immed, Rs

# This is an Immediate format instruction
# Mem[Reg[rs]] = unsigned immediate;
```

Modify the datapath and the control table to implement the SWUI instruction. Use the minimal amount of additional hardware and fill/modify/expand the control table. **Mark the full datapath for the new instruction.**



Type	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp	Jump	SWUI
R	1	0	0	1	0	0	0	10	0	X
lw	0	1	1	1	1	0	0	00	0	X
sw	X	1	X	0	0	1	0	00	0	0
beq	X	0	X	0	0	0	1	01	0	X
j	X	X	X	0	0	0	X	XX	1	X
SWUI	X	10	X	0	0	1	0	00	0	1

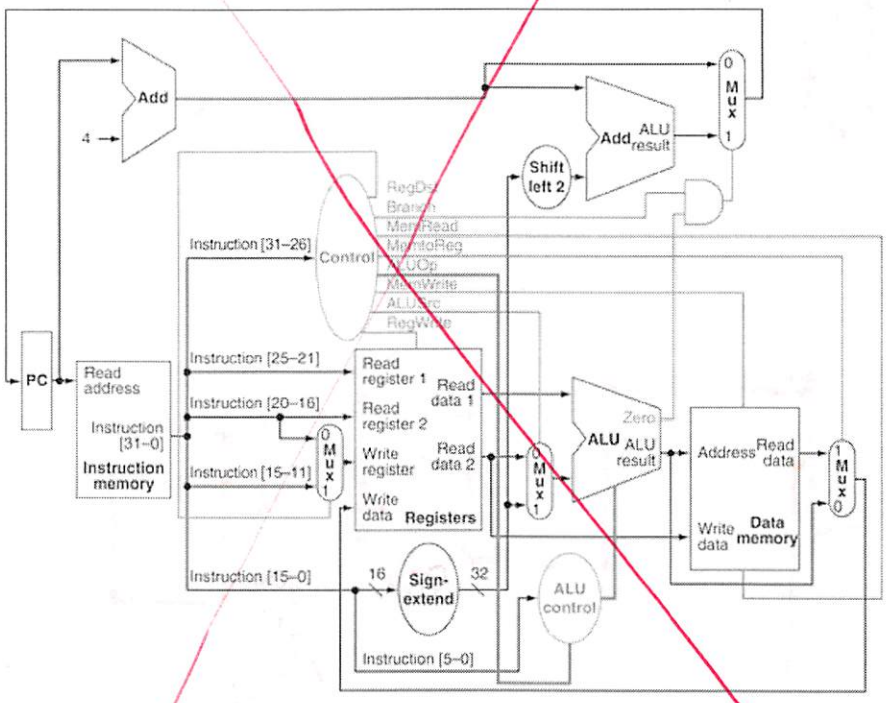
9 [15 pts] Single-cycle Datapath

Implement a "add on equal" instruction into the single-cycle MIPS datapath.

```
ADDE Rs, Rt, Rd

#if Reg[Rs] == 2's complement value in shamt field
#   Reg[Rd] = Reg[Rs] + Reg[rt];
```

Modify the datapath and the control table to implement the instruction. Use the minimal amount of additional hardware and fill/modify/expand the control table. **Mark the full datapath for the new instruction.**



Type	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp	Jump
R	1	0	0	1	0	0	0	10	0
lw	0	1	1	1	1	0	0	00	0
sw	X	1	X	0	0	1	0	00	0
beq	X	0	X	0	0	0	1	01	0
j	X	X	X	0	0	0	X	XX	1

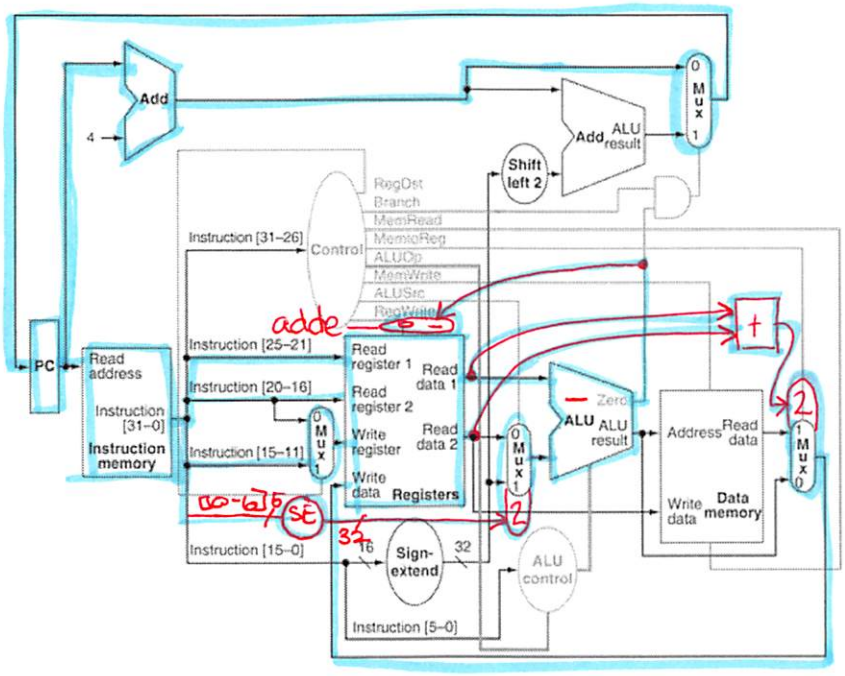
9 [15 pts] Single-cycle Datapath

Implement a "add on equal" instruction into the single-cycle MIPS datapath.

```
ADDE Rs, Rt, Rd

#if Reg[Rs] == 2's complement value in shamt field
#   Reg[Rd] = Reg[Rs] + Reg[rt];
```

Modify the datapath and the control table to implement the instruction. Use the minimal amount of additional hardware and fill/modify/expand the control table. **Mark the full datapath for the new instruction.**



Type	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp	Jump	adde
R	1	0	0	1	0	0	0	10	0	0
lw	0	0 1	0 1	1	1	0	0	00	0	0
sw	X	0 1	X X	0	0	1	0	00	0	0 } must be zero
beq	X	0 0	X X	0	0	0	1	01	0	0 }
j	X	X X	X X	0	0	0	X	XX	1	0 }
adde	1	10	10	0	0	0	0	01	0	1

NAME _____
First Last

Student ID# _____

STONY BROOK UNIVERSITY

COMPUTER SCIENCE DEPARTMENT

MIDTERM EXAMINATION
VERSION B

CSE 320
Fall 2014
October 14, 2014

This is a closed-book exam. (80 minutes)
Use this form for your work and return it.

The exam has 8 problems.
It is crucial to show all work done on the provided paper.

#1 _____ (13 pts)
#2 _____ (10 pts)
#3 _____ (12 pts)
#4 _____ (12 pts)
#5 _____ (8 pts)
#6 _____ (10 pts)
#7 _____ (8 pts)
#8 _____ (12 pts)
#9 _____ (15 pts)

TOTAL _____ (100 Max)

1 [13pts] Short Answer

(a) [3 pts] What is the difference between a decoder and an encoder?

- Same -

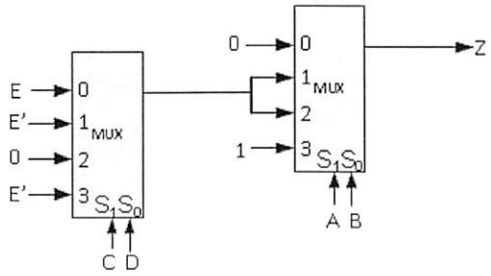
(b) [3 pts] Draw and explain the operation of a 2-bit selector demultiplexor?

- Same -

(b) [3 pts] What determines the minimum clock cycle time of a circuit?

- Same -

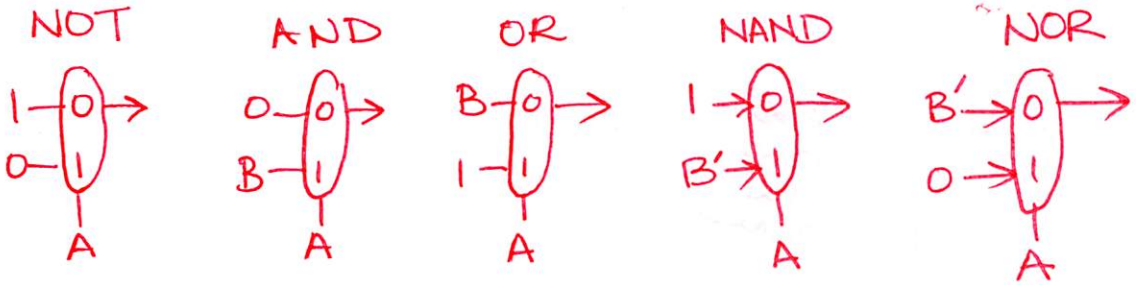
(d) [4 pts] What is the Boolean equation for the following circuit? Do not minimize.



$AB + (AB' + A'B)(C'D'E + C'DE' + CDE') = Z$

2 [10 pts] Short Answer

(a) [5 pts] Demonstrate how a 2-bit multiplexor is a universal gate.



{NOT, AND}
{NOT, OR}
{NAND} {NOR}

(b) [5 pts] Demonstrate how NOR is a universal gate.

Not

OR

And

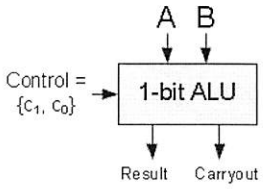
3 [12 pts] ALUs

(a) [4 pts] Using 1-bit ALUs we make an 8-bit ALU (operates on 8 bit numbers) and a 16-bit ALU (operates on 16-bit numbers). Consider A AND B, A + B (addition) and A OR B performed by these ALUs.

Fill in the blank with "less", "same", or "more"

(i) An 8-bit ALU requires _____less_____ time for A + B as a 16-bit ALU.
(ii) A 16-bit ALU requires _____same_____ time for A AND B than an 8-bit ALU.
(iii) An 8-bit ALU requires _____more_____ time for A + B than a 16-bit ALU doing A AND B.
(iv) A 16-bit ALU requires _____same_____ time for A AND B than an 8-bit ALU doing A OR B

(b) [8 pts] Using 2-bit (1-selector) multiplexors and 1-bit Adders, build a 2-bit ALU as described by the figure (with $Carry_{out}$ and Zero output signals) which performs the following functions. Numbers are in two's complement form. Note: There is no $Carry_{in}$ signal to the larger unit.



Control	Operation
00	A + B
01	A - B
10	B - A
11	A AND B

- See attached sheet -

Ver B

4 [12 pts] Boolean Expressions.

$$g = (A + B)' + A'BC' + (A + C')'$$

(a) [2 pts] How many literals appear in the boolean expression? 7

(b) [4 pts] Find all minterms, answer in boolean expression form:

$(A'B') + A'BC' + (A'C)$

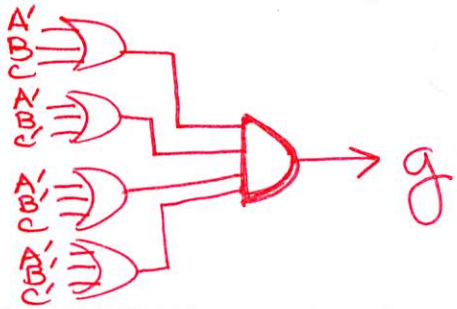
$A'B'C' + A'B'C + A'BC' + A'BC + \cancel{A'B'C}$ Duplicate

0 1 2 3

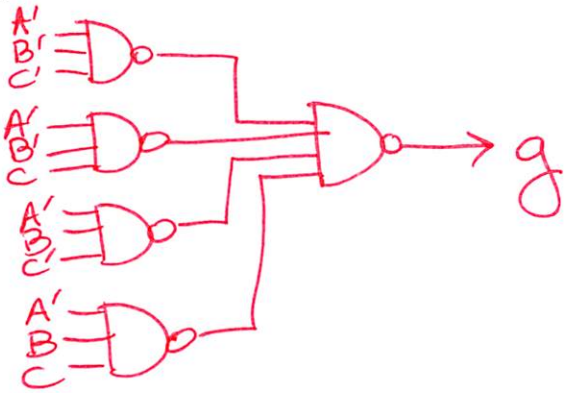
(c) [2 pts] Name the maxterms, answer in $\prod M(?)$ form :

$\prod M(4,5,6,7)$

(d) [2 pts] Draw the 2-level AND-OR or OR-AND gate network for part (c). Do not minimize the expression.



(e) [2 pts] Draw the 2-level NAND-NAND or NOR-NOR gate network for part (b). Do not minimize the expression.



5 [8 pts] Using the postulates of Boolean algebra, minimize the following:

$$((A' + B')' + A'B')(C'D' + CD) + (AC)' = A' + C' + BD$$

$ABC'D' + ABCD + A'B'C'D' + A'B'CD + A' + C'$

$A'(B'C'D' + B'CD + 1) + C'(1 + ABD') + ABCD$

$A' + C' + ABCD$

$A' + (C' + ABCD)$

$A' + (C' + ABD)$

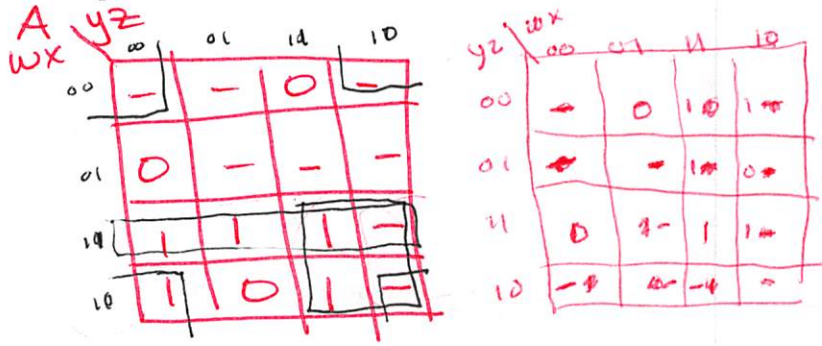
$(A' + ABD) + C'$

$A' + BD + C'$

6 [10 pts] Consider the following truth table for A:

w	x	y	z	A
0	0	0	0	-
0	0	0	1	-
0	0	1	0	-
0	0	1	1	0
0	1	0	0	0
0	1	0	1	-
0	1	1	0	-
0	1	1	1	-
1	0	0	0	1
1	0	0	1	0
1	0	1	0	-
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	-
1	1	1	1	1

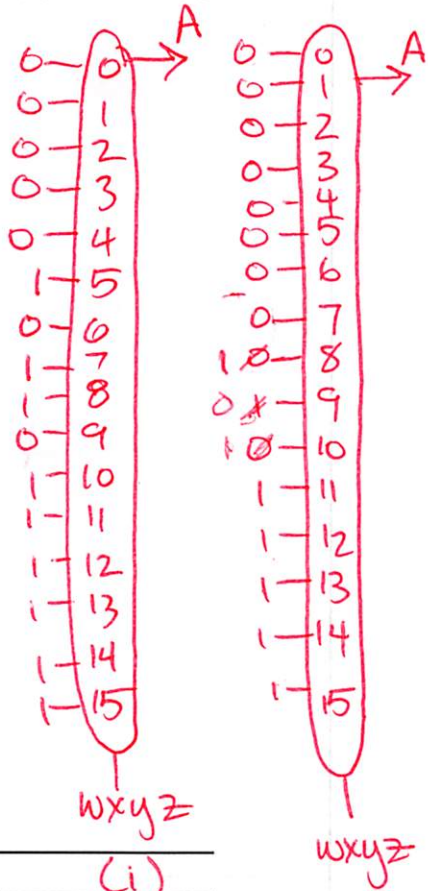
(a) [2 pts] Draw the k-map for A:



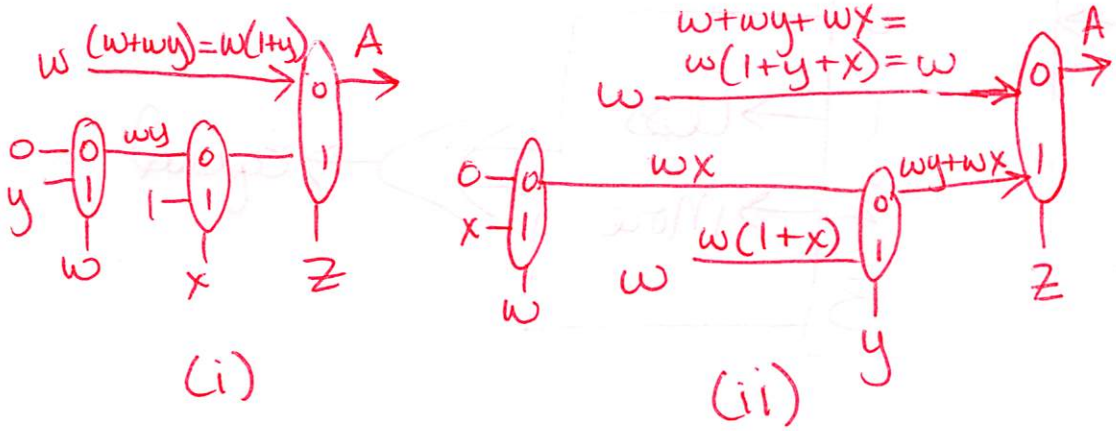
(b) [2 pts] Write the minimal SOP expression for A:

$wz' + wy + xz$ (i) or $wz' + wy + wx$ (ii) $x'z' + wx + wy$ (iii)

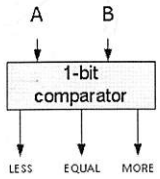
(c) [2 pts] Implement the expression from part (b) with a single 4-bit selector mutiplexor



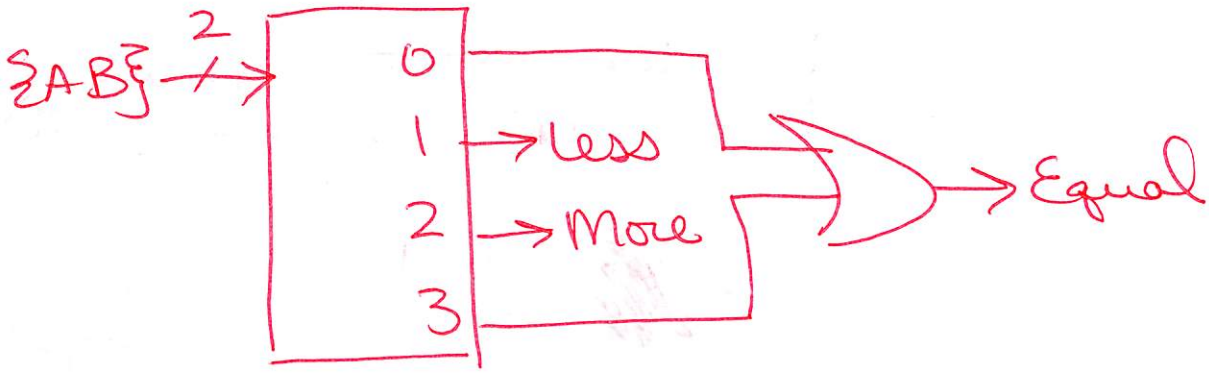
(d) [4 pts] Implement the expression from part (b) with a 2-bit mutiplexors



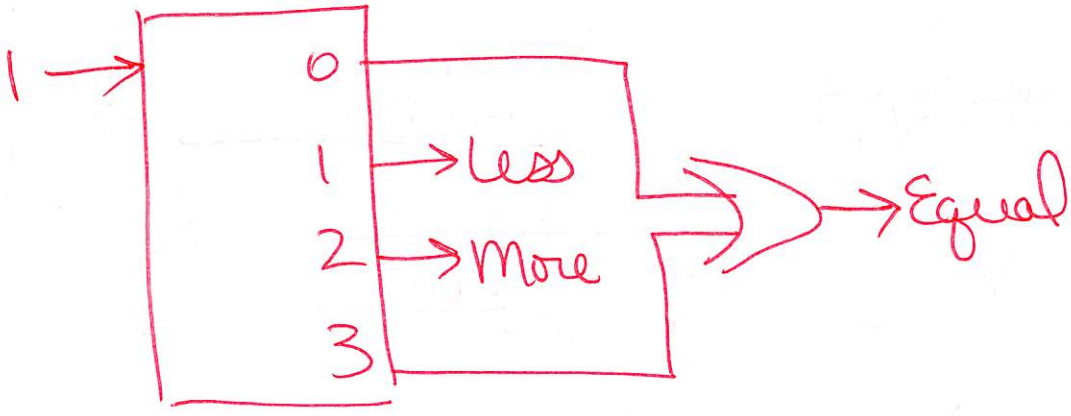
7 [8 pts] Consider the 1-bit comparator unit as specified below.
LESS = 1 when A < B, EQUAL = 1 when A == B, MORE = 1 when A > B



(a) [4 pts] Implement the unit using a 2-bit decoder and OR gates.



(b) [4 pts] Implement the unit using a 2-selector demultiplexor and OR gates.



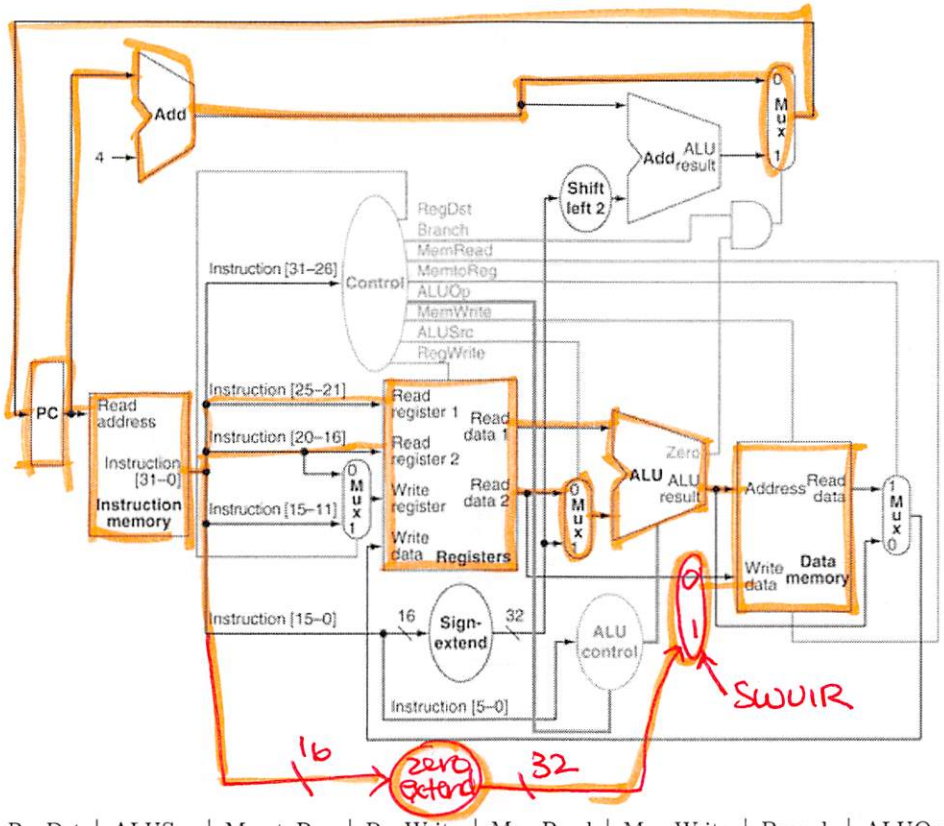
8 [12 pts] Single-cycle Datapath

Implement a "store word unsigned immediate via register" instruction into the single-cycle MIPS datapath.

```
SWUIR immed, Rs, Rt

# This is an Immediate format instruction
# Mem[Reg[rs] + Reg[rt]] = unsigned immediate;
```

Modify the datapath and the control table to implement the SWUIR instruction. Use the minimal amount of additional hardware and fill/modify/expand the control table. **Mark the full datapath for the new instruction.**



Type	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp	Jump	SWUIR
R	1	0	0	1	0	0	0	10	0	X
lw	0	1	1	1	1	0	0	00	0	X
sw	X	1	X	0	0	1	0	00	0	0
beq	X	0	X	0	0	0	1	01	0	X
j	X	X	X	0	0	0	X	XX	1	X
SWUIR	X	X	X	0	0	1	0	00	0	1

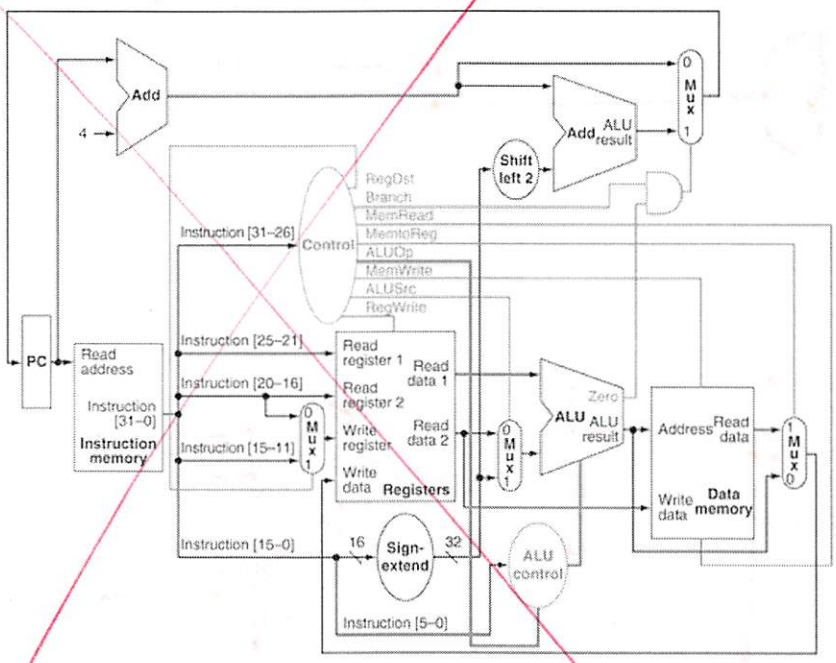
9 [15 pts] Single-cycle Datapath

Implement a "add on not equal"instruction into the single-cycle MIPS datapath.

```
ADDNE Rs, Rt, Rd

#if Reg[Rs] != 2's complement value in shamt field
#   Reg[Rd] = Reg[Rs] + Reg[rt];
```

Modify the datapath and the control table to implement the instruction. Use the minimal amount of additional hardware and fill/modify/expand the control table. **Mark the full datapath for the new instruction.**



Type	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp	Jump
R	1	0	0	1	0	0	0	10	0
lw	0	1	1	1	1	0	0	00	0
sw	X	1	X	0	0	1	0	00	0
beq	X	0	X	0	0	0	1	01	0
j	X	X	X	0	0	0	X	XX	1

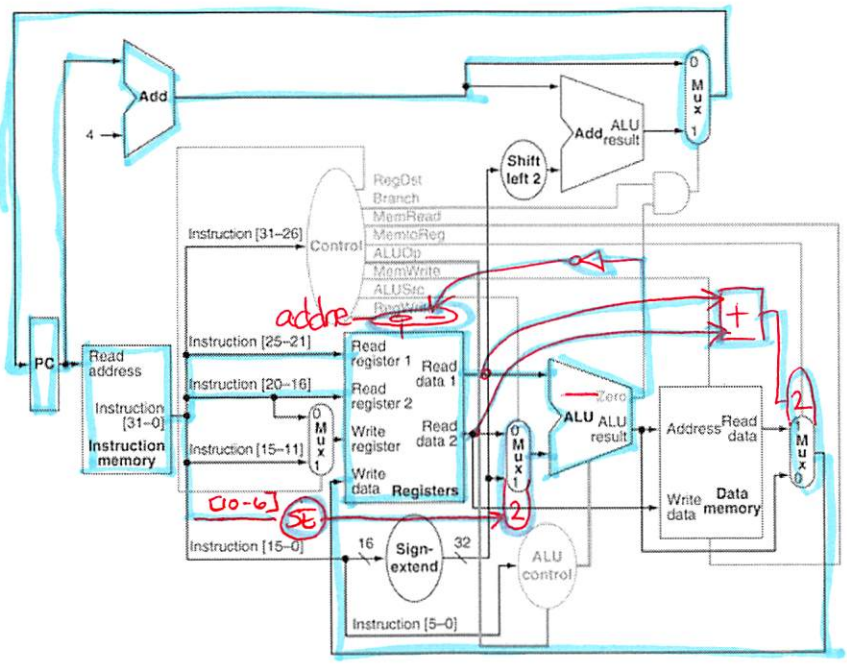
9 [15 pts] Single-cycle Datapath

Implement a "add on not equal"instruction into the single-cycle MIPS datapath.

```
ADDNE Rs, Rt, Rd

#if Reg[Rs] != 2's complement value in shamt field
#   Reg[Rd] = Reg[Rs] + Reg[rt];
```

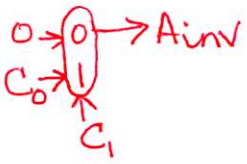
Modify the datapath and the control table to implement the instruction. Use the minimal amount of additional hardware and fill/modify/expand the control table. **Mark the full datapath for the new instruction.**

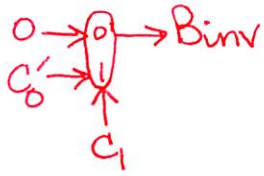


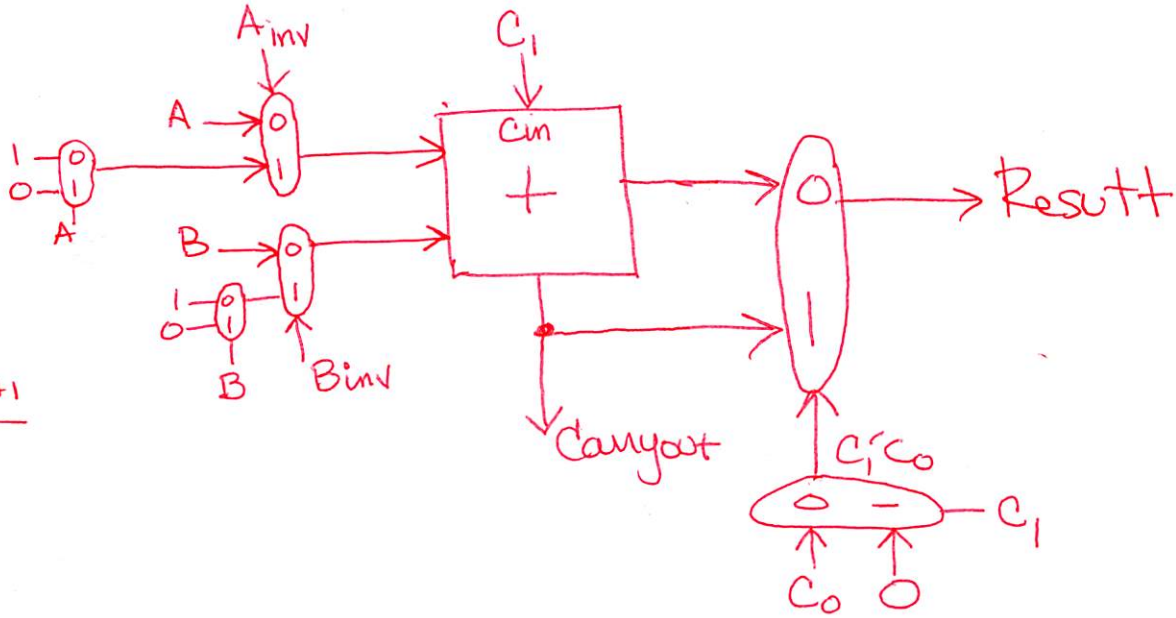
Type	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp	Jump	addne
R	1	0	0	1	0	0	0	10	0	0
lw	0	1	1	1	1	0	0	00	0	0
sw	X	1	X	0	0	1	0	00	0	0
beq	X	0	X	0	0	0	1	01	0	0
j	X	X	X	0	0	0	X	XX	1	0
addne	1	10	10	0	0	0	0	01	0	1

must be zero

VER. A







C_1C_0	Ainv	Binv	2's Comp+1
00	0	0	0
01	0	0	0
10	0	1	1
11	1	0	1

-1 4 bit selector

-1 ▽1 adder

-1 Cout wrong

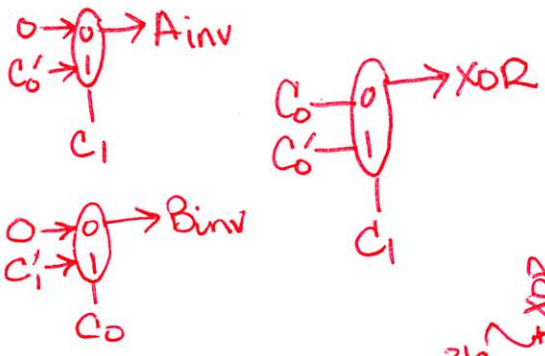
~~-1 Binv~~

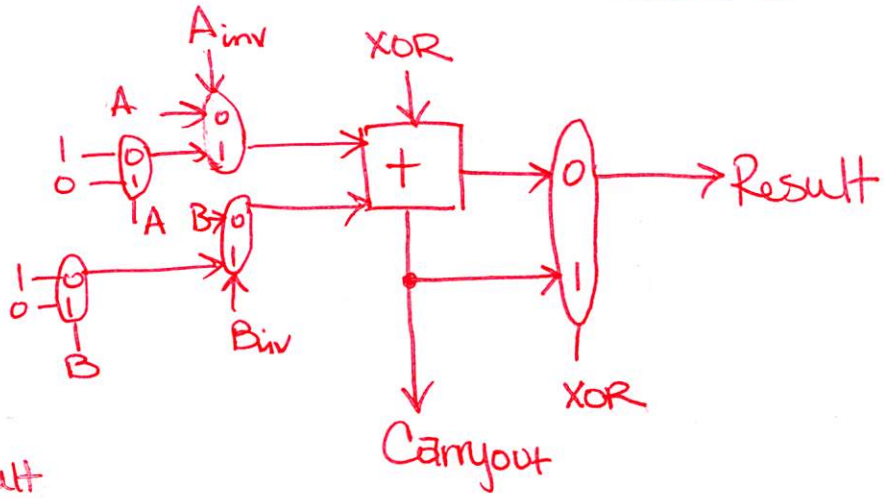
-1 wrong Operation

-1 no ~~inputs~~ Cin to adders

-1 invalid gates

VER. B





C_1C_0	Ainv	Binv	2'S Comp+1	Result
00	0	0	0	0
01	0	1	1	1
10	1	0	1	1
11	0	0	0	0