Sample Exam 2

10-Nov-18

16 November 2018

__________________ , _____________

Last Name , First Name

Instructions:

- ***Show all work* on the *front* of the test papers. No work shown may mean 0 points given!** *If you need more room, make a clearly indicated note on the front of the page, "MORE ON BACK", and use the back. The back of the page will **not** be graded without an indication on the front.*
- ***Read** each question **carefully** and **follow the instructions**. Unless otherwise stated, **you must show your work and clearly illustrate your steps**.*
- *If you round a numerical answer, you must give at least **3 significant digits**.*
- *Put your name at the top of **each** test page and be sure your exam consists of the number of pages designated in the headers.*
- *The space provided does **NOT** necessarily represent the amount of writing necessary.*
- *You may **not** use any notes, homework, labs, or other books. Only **memoryless** calculator is allowed.*
- ***You may not use any notes, homework, labs, or other books.***

COMMENTS, FEEDBACK, or any special instructions for the professor:

Problem	Available	Points
1		
2		
3		
4		
5		
Total		

Important: *In completing this exam, I used a calculator with no communications capability, and no information of relevance to the course was stored in the calculator. I did not use any other electronic device or any other references. My work was solely my own.*

Your Calculator's Maker and Model#: ________________, **Signature:** ___________________

You must sign this to receive credit for the exam.

Important Boolean Algebra Laws

Identity law

- A + 0 = A
- A·1 = A

Zero and One laws

- A + 1 = 1
- A·0 = 0

Associative laws

- A + (B + C) = (A + B) + C
- A· (B·C) = A· (B·C)

Distributive laws

- (B + C) = (A.B) + (B.C)
- A+(B·C) = (A+B) · (A+C)

DeMorgan's laws

- (A·B)'=A'+B'
- (A+B)'=A'·B'

Inverse laws

- A + A' = 1
- A·A' = 0

Commutative laws

- A + B = B + A
- A·B = B·A

__________________ , _____________

Last Name , First Name

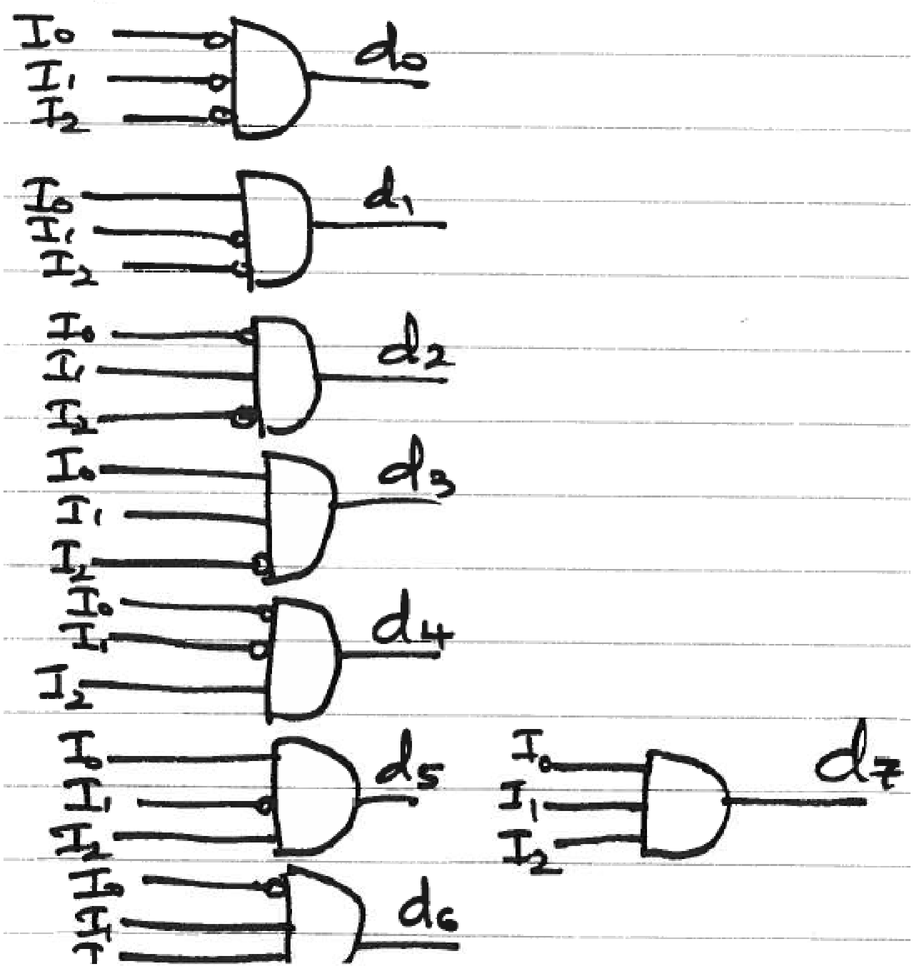
Remember to show ALL work here and in EVERY problem on this exam.

Q1. What is a decoder? Construct a 3-to-8 decoder with basic gates. Clearly show all your steps with the truth table.

ANS:

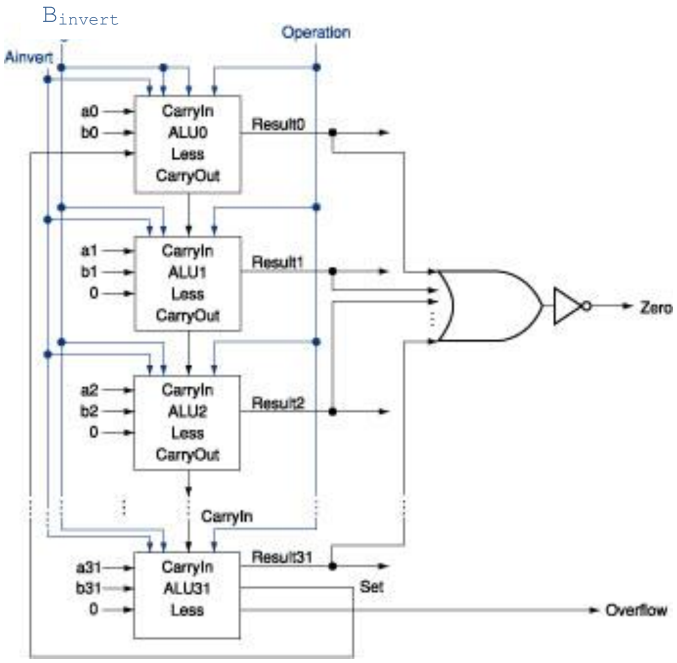
A decoder is a combinational circuit that has ‘n’ inputs and ‘2^n’ outputs.

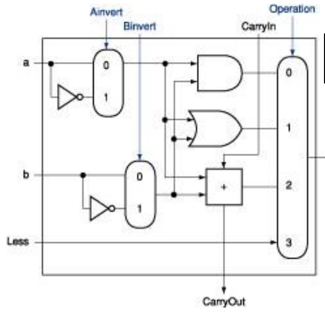
I2	I1	I0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

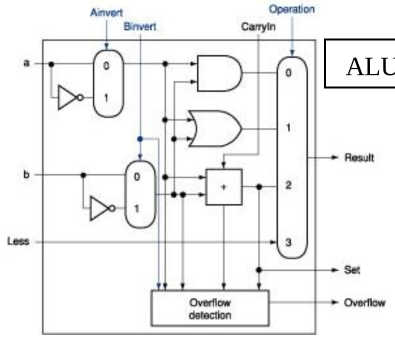


________________ , ____________
Last Name , First Name

Q2. *The 32-bit ALU diagram we covered in class is attached below:*







The ALU line control code is [A$_{invert}$ (1 bit), B$_{invert}$ (1 bit), Operation (2 bit)]. Answer the following questions:

a) Give ALU line code for OR function

Solution

0001

b) Give ALU line code for SUB (subtraction) function

Solution

0110

Sample Exam 2

16 November 2018

10-Nov-18

________________ , ____________

Last Name , First Name

c) Assume

a= `00000000000000001111111111111111` and
b = `10000001110000000000000000000000`.

Determine the ALU output after the ALU control code [0011] is executed. You need to give your reasoning which does not necessarily mean long calculation.

(**Note:** You don’t have to give your result in the 32 bit format. If you can write your result with decimal number, it will be fine)

Solution

Since a+b<0, ALU result is 1

d) Under what kind of situation, this ALU will give wrong result for its SLT operation. Assume every component in this ALU is in good condition. (**Note:** when ALU executes SLT command, if A<B, then output should be 1)

Solution

When overflow happens

Sample Exam 2

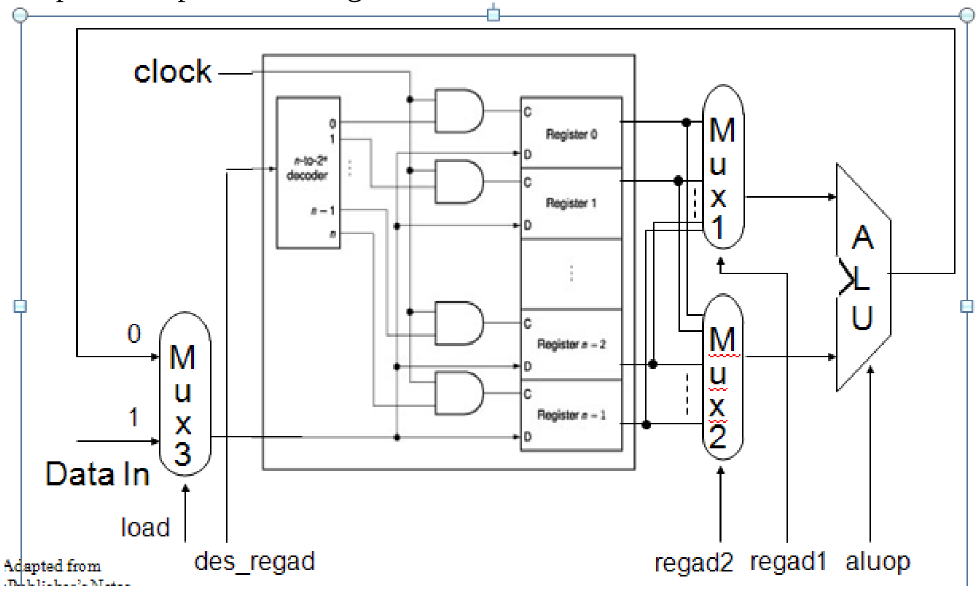
10-Nov-18

16 November 2018

________________ , ____________

Last Name , First Name

Q3. A simple microprocessor diagram is illustrated below:



This processor's machine code format is [load, regad1, regad2, des_regad, aluop], where

load signal: 1 (read from external input), 0 (read from ALU)
regad1: the address of the register for the ALU input 1
regad2: the address of the register for the ALU input 2
des_regad: the address of the destination register which data is written into
aluop: the ALU function selection
Register number `r0~r15` (16 registers each of which has 16 bits)

ALU op code	**ALU Function**	**Assembly Code**
0 (000)	And	`and`
1 (001)	Or	`or`
2 (010)	NAND	`nand`
3 (011)	NOR	`nor`
4 (100)	XOR	`xor`
5 (101)	Addition	`add`
6 (110)	Subtraction	`sub`
7 (111)	Slt	`slt`

The syntax of assembly language of our simple processor is defined as following:

'`sub r3, r1, r2`' means: r3=r1–r2, Machine Code: [`0000100100011110`]

'`slt r3, r1, r2`' means:

If r1<r2 ⇒r3 =1

Otherwise, r3=0, Machine Code: [`0000100100011111`]

'`load r5`' means: external input data ⇒ r5, Machine Code: [`1XXXXXXXX0101XXX`] (**we assign all of 'X' equal to 0**)

Based on the information above, answer the following questions:

Sample Exam 2

16 November 2018

________________ , ____________

Last Name , First Name

a) (True or False)

[T] In this processor, during each clock cycle, data from either ALU output or external data input will be written into one of its registers.

b) Translate the assembly code '`add r1, r2, r3`' to a machine code.

Solution

0001000110001101

c) Translate the machine code [0110000111110011] back to an assembly code.

Solution

```
NOR, r14, r12, r3
```

d) Someone writes a two line program for this computer as following:

```
0001100110011111
0001100110011010
```

Determine the content of register `r3` as a 16 bit binary sequence after this program is executed assuming that each register has 16 bits.

Solution

```
These two lines are
```

slt r3, r3, r3 # set r3=0
nand, r3, r3, r3 #r3=1111111111111111

______________ , ____________

Last Name , First Name

Q4. The following problem deal with transition from C to MIPS. Assume that the variables f, g, h, i, and j are assigned to registers $s0, $s1, $s2, $s3, and $s4, respectively. Assume that the base address of the arrays A and B are in registers $s6 and $s7, respectively.

a.	f = -g – A[6];
b.	B[4] = A[i - j];

For the C statements above, (i) what is the corresponding MIPS assembly code? (ii) How many different registers are needed to carry out the C statement? (Need to work on case **a** and **b**)

(i)

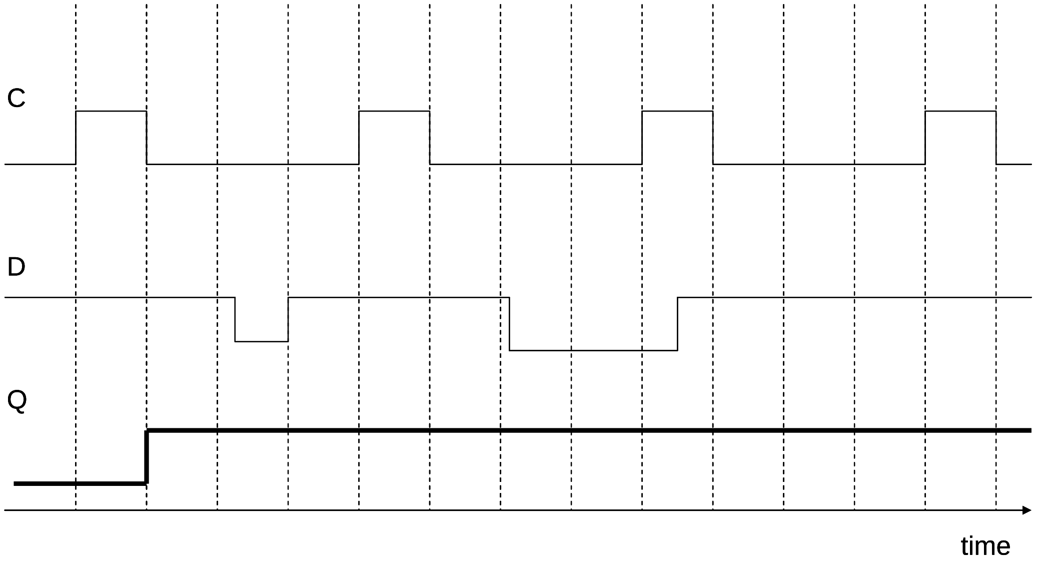
a.
```
lw $s0, 24($s6)
sub $s0, 0, $s0
sub $s0, $s0, $s1
```

b.
```
sub $t0, $s3, $s4
sll $t0, $t0, 2
add $t0, $s6, $t0
lw $t1, 0($t0)
sw $t1, 16($7)
```

(ii) a. 3 and b. 6

Q5. A D flip-flop output diagram is illustrated below. Assuming that (i) the initial value of Q is 0 and (ii) setup and hold time of D-type flip flop is zero, then, based on the clock and input waveform below, plot its output (Q) waveform.

ANS:



Sample Exam 2

10-Nov-18

16 November 2018

____________________ , ______________

Last Name , First Name

MIPS Instruction Formats:

Name	**Fields**					
Field Size	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
R-Format	opcode	rs	rt	rd	shamt	funct
I-Format	opcode	rs	rt	address/immediate		
J-Format	opcode	target address				

MIPS Instructions:

Instruction	*Type*	*opcode/funct (hex)**	*Example*	*Meaning*
add	R	0/20	add $s1, $s2, $s3	R[rd]=R[rs]+R[rt]
addi	I	8	addi $s1, $s2, 100	R[rt]=R[rs]+SignExImm
and	R	0/24	and $s1, $s2, $s3	R[rd]=R[rs]&R[rt]
andi	I	c	andi $s1, $s2, 100	R[rt]=R[rs]&ZeroExImm
beq	I	4	beq $s1, $s2, 25	If (R[rs]==R[rt]) PC=PC+4+Imm*4
bne	I	5	bne $s1, $s2, 25	If (R[rs]!=R[rt]) PC=PC+4+Imm*4
j	J	2	j 2500	PC=jumpAddr[1]
jal	J	3	jal 2500	R[31]=PC+4; PC=jumpAddr[1]
jr	R	0/8	jr $ra	PC=R[rs]
lui	I	f	lui $s1, 100	R[rt]={imm, 16 of 0s}
lw	I	23	lw $s1, 100($s2)	R[rt]=MEM[R[rs]+ SignExImm]
nor	R	0/27	nor $s1, $s2, $s3	R[rd]=~(R[rs]|R[rt])
or	R	0/25	or $s1, $s2, $s3	R[rd]=R[rs]|R[rt]
ori	I	d	ori $s1, $s2, 100	R[rt]=R[rs]|ZeroExImm
sll	R	0/0	sll $s1, $s2, 10	R[rd]=R[rt]<<shamt
slt	R	0/2a	slt $s1, $s2, $s3	R[rd]=(R[rs]<R[rt])? 1:0
slti	I	a	slti $s1, $s2, 100	R[rt]=(R[rs]<SignExImm)? 1:0
srl	R	0/2	srl $s1, $s2, 10	R[rd]=R[rt]>>shamt
sub	R	0/22	sub $s1, $s2, $s3	R[rd]=R[rs]-R[rt]
sw	I	2b	sw $s1, 100($s2)	MEM[R[rs]+ SignExImm]= R[rt]

MIPS Register Convention:

Use	**Number**	**Name**
The constant 0	$0	$zero
Reserved for Assembler	$1	$at
Return Values	$2-$3	$v0-$v1
Arguments	$4-$7	$a0-$a3
Temporary	$8-$15	$t0-$t7
Saved Temporaries	$16-$23	$s0-$s7
Temporary	$24-$25	$t8-$t9
Used by Kernel	$26-$27	$k0-$k1
Global Pointer	$28	$gp
Stack Pointer	$29	$sp
Frame Pointer	$30	$fp
Return Address	$31	$ra

__________________ , ______________

Last Name , First Name

This Page Intentionally Left Blank