Midterm Review

(Review Sessions)

Computer Architecture and Organization

School of CSEE

P1

- What is the “von Neumann architecture”? What is the benefits? Explain in your words. (5points)

input + CPU + ALU + memory + output

P2

- Why we use the saved registers ($s0~$s7 in MIPS)? How to handle the saved registers in programmer's view?

=) 호출규약 참조.

P3

- What is the value in the register $v0 after the following sequence of instructions are executed? Assume $a0 is 10 initially.

```
Begin:   addi   $t0,   $zero,   0
         addi   $t1,   $zero,   1
Loop:    slt    $t2,   $t1,     $a0
         beq    $t2,   $zero,   Finish
         add    $t0,   $t0,     $t1
         addi   $t1,   $t1,     2
         j      Loop
Finish:  add    $v0,   $t0,     $a0
```

35

$t_0 += t_1$

$t_1 += 2$

t_0	t_1
0	1
1	3
4	5
9	7
16	9
25	11

35 25 + 10

P4~P9

- For P4 ~ P9, consider the following C code. Assume that x and y are 3x3 matrix.

main.c	calc.c

void matrix_diff(int x[][], int y[][])
{
  int i, j;
  for (i = 0; i != 3; i++)
  for (j = 0; j != 3; j++)
  x[i][j] = diff(x[i][j], y[i][j]);
  ...
}
…
	int diff(int xx, int yy)
	{
	  if (xx >= yy)
	  return xx – yy;
	  else
	  return yy – xx;
	}
	…

P4

- Explain the procedure how above C codes are executed in the computer. Assume the static linking.

P5

- Below MIPS code is excutable images from two C codes. Fill in the empty MIPS codes
  - Assumptions: i in $s0, j in $s1, x[][] in $a0, y[][] in $a1, xx in $a2, yy in $a3

```
for (i = 0; i != 3; i++)
    for (j = 0; j != 3; j++)
        x[i][j] = diff(x[i][j], y[i][j]);

if (xx >= yy)
  return xx – yy;
else
  return yy – xx;
```

2	$0040\ 1000_{hex}$	matrix_diff:	li	$t0,	3	
3	$0040\ 1004_{hex}$		li	$s0,	0	
4	$0040\ 1008_{hex}$	loop1:	li	$s1,	0	
5	$0040\ 100c_{hex}$	loop2:				
6	$0040\ 1010_{hex}$					
7	$0040\ 1014_{hex}$					
8	$0040\ 1018_{hex}$		add	$t2	$a0,	$t1
9	$0040\ 101c_{hex}$		add	$t3,	$a1,	$t1
10	$0040\ 1020_{hex}$		lw	$a2,	0($t2)	
11	$0040\ 1024_{hex}$					
12	$0040\ 1028_{hex}$		jal	diff		
13	$0040\ 102c_{hex}$		sw	$v0,	0($t2)	
14	$0040\ 1030_{hex}$					
15	$0040\ 1034_{hex}$					
16	$0040\ 1038_{hex}$					
17	$0040\ 103c_{hex}$					
18	…		….			
19	$0040\ c000_{hex}$	diff:	slt	$t5,	$a2,	$a3
20	$0040\ c004_{hex}$					
21	$0040\ c008_{hex}$					
22	$0040\ c00c_{hex}$		jr	$ra		
23	$0040\ c010_{hex}$	ifdiff	sub	$v0,	$a3,	$a2
24	$0040\ c014_{hex}$		jr	$ra		

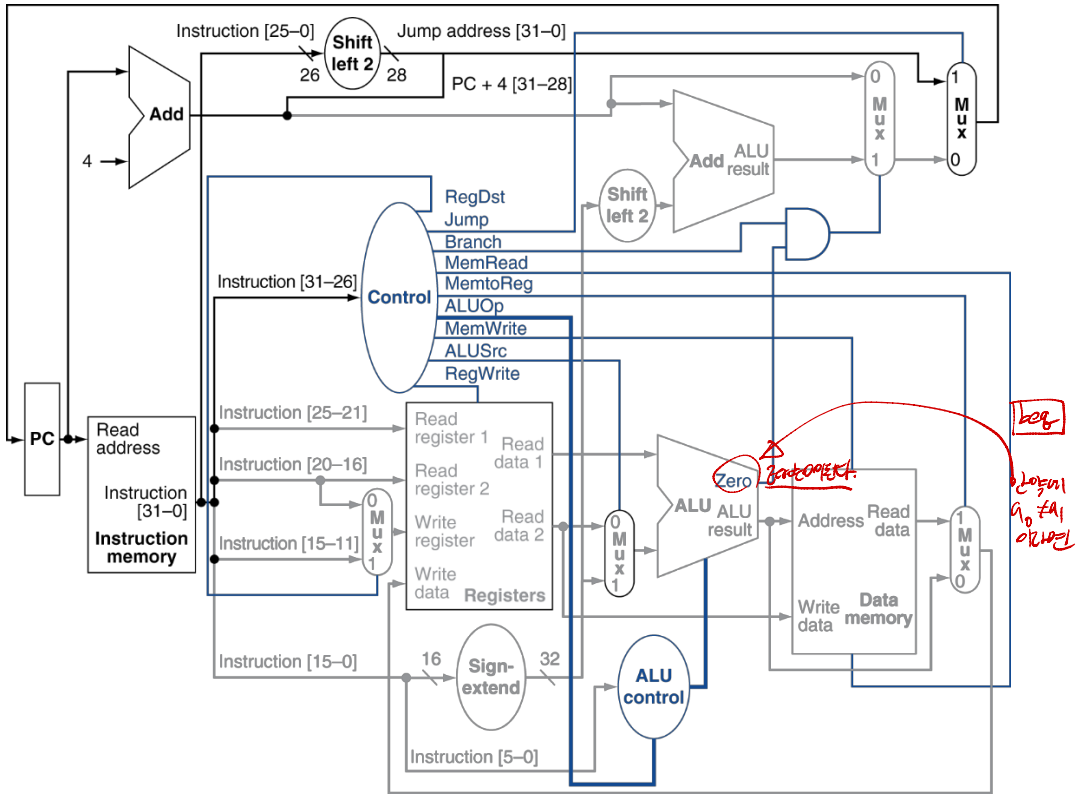
P6

- Encode line 12 (i.e., *jal*) and 17 (i.e., *bne*) into machine code. When you write your answers, use hexadecimal for each field.

bne $s0,

P7

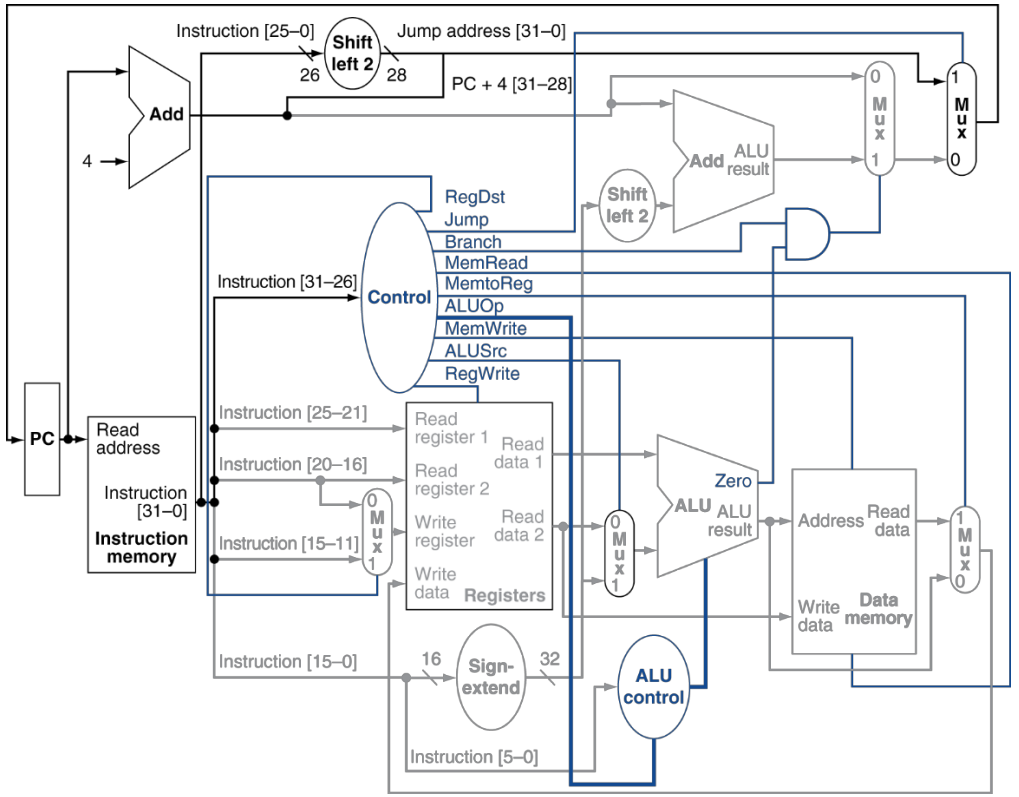
- For the instruction *bne* (branch not equal), add extra datapaths and control signals to figure below, if necessary. If not, just say "No need of extra datapath". Also determine the value of control signals for the instruction *bne*. Do not change ALU.



ALUOp(2bit)	Instructions
00	lw, sw
01	beq
10	Arithmetic

P8

- Similar to P7, consider the instruction *jr* (Jump Register). You should give datapath and control signals if necessary. (You don't need to write the extra control signals you added in P7.)



P9

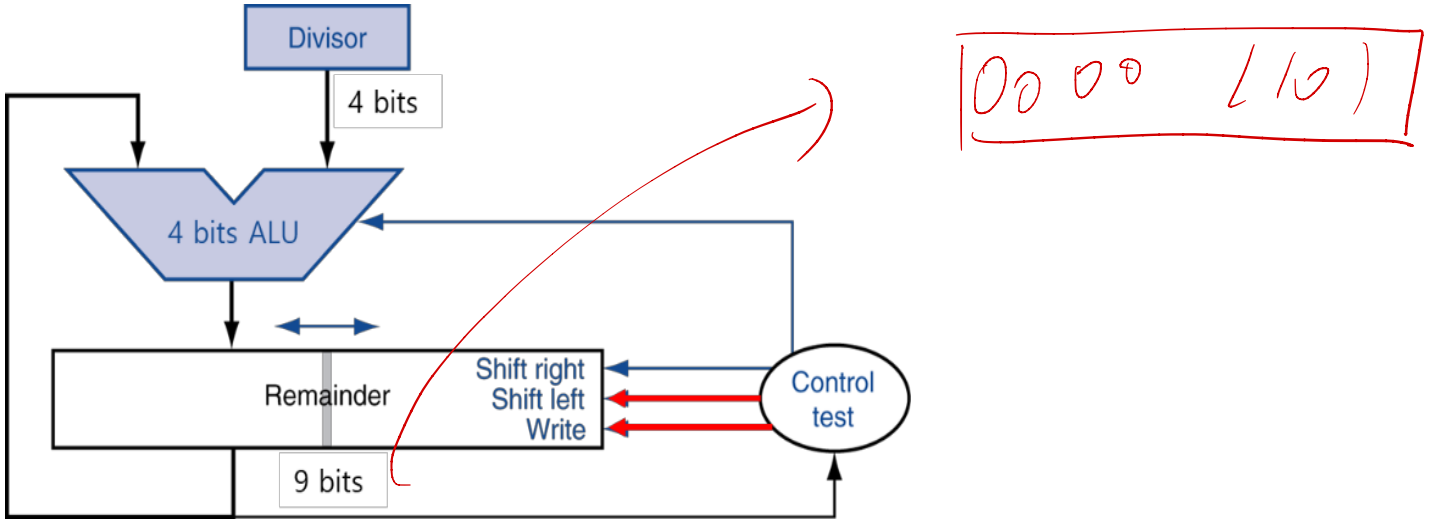
- Assume that the pipelined datapath executes the procedure "*diff*" (line 19~24). List all possible hazards. Also explain the reasons and simple solutions (You don't need to draw datapath for your solution. Assume that our pipeline datapath covers all instructions in the procedure "diff").

P10

- Suppose that the instruction slt in procedure “diff” (line 19) is fetched from memory for clock 20th. If $a2 is 30 and $a3 is 20, which instruction is in instruction decoding stage for clock 25th? Consider it with branch not taken.

P11

- Using the hardware shown in below, divide unsigned 4-bit integers 11ten (Dividend) by 3ten (Divisor). You should show the contents of each register on each step.



0000 (10)

P12

- Add 1.1010×2^{-4} and 1.0110×2^{1}, assuming that 5 significant bits, 1 guard, 1 round, and 1 sticky bit. Please show the calculation procedure on each step. (Hint! Round up if more than half of the least significant bit)