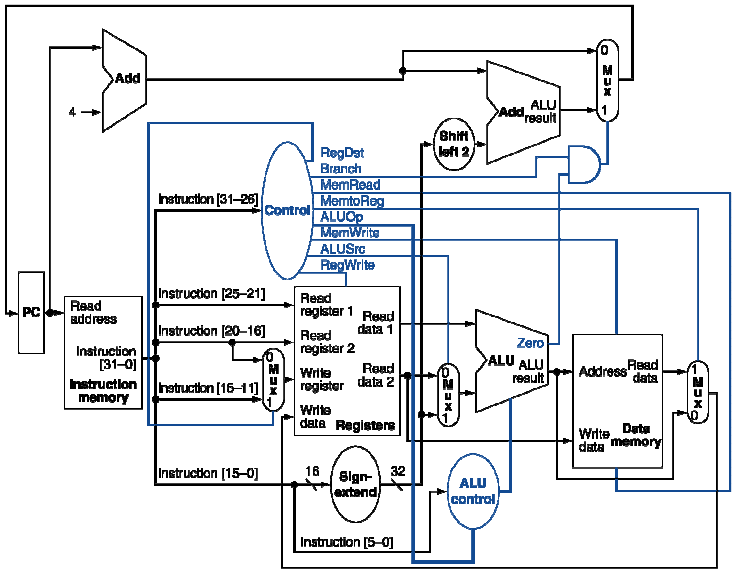
Midterm of Computer Organization
November 19, 2009

1. (15%) Consider a simple single-cycle implementation of MIPS ISA. The operation times for the major functional components for this machine are as follows:

Component	Latency
ALU	10 ns
Adder	8 ns
ALU Control Unit	2 ns
Shifter	3 ns
Control Unit/ROM	4 ns
Sign/zero extender	3 ns
2-1 MUX	2 ns
Memory (read/write) (instruction or data)	15 ns
PC Register (read action)	1 ns
PC Register (write action)	1 ns
Register file (read action)	7 ns
Register file (write action)	5 ns
Logic (1 or more levels of gates)	1 ns

Below is a copy (from textbook) of the MIPS single-cycle datapath design. Suppose that all instructions have the same instruction fetch and decode steps. The critical paths for the different instruction types that need to be considered are: R-format, Load-word, and store-word.



(a) In the table below, indicate the components that determine the path delay for the respective instruction, in the order that the critical path occurs. If a component is used, but not part of the critical path of the instruction (ie happens in parallel with another component), it should not be in the table. The register file is used for reading and for writing; it will appear twice for some instructions.

Instruction Type	Hardware Elements Used By Instruction											
R-Format												
Load												
Store												

(b) Suppose that all instructions begin by reading the PC register with a latency of 2ns. Please place the latencies of the components that you have decided for the critical path of each instruction in the table below. Compute the sum of each of the component latencies for each instruction.

Instruction Type	Hardware Latencies For Respective Elements										Total
R-Format	2 ns										
Load	2 ns										
Store	2 ns										

(c) For this MIPS processor, what will be the resultant clock cycle time? And, what frequency will the machine run?

2. (15%) Translate the following C codes into MIPS instructions, assuming that `compare` is the first function called:

```
int compare(int a, int b){
    if ( sub(a,b) >= 0)
        return 1;
    else
        return 0;
}
int sub(int a, int b){
    return a-b;
}
```

Be sure to handle the stack and frame pointers appropriately. The variable code font is allocated on the stack. Draw the status of the stack before calling *compare* and during each function call. Indicate the names of registers and variables stored on the stack and mark the location of `$sp` and `$fp`.

3. (10%) The following table shows the number of instructions for a program.

Arith	Store	Load	Branch	Total
500	50	100	50	700

(a) Assuming that Load instruction takes 5 cycle, Arith and Store 4 cycles and Branch 3 cycles, what is the execution time of the program running in 2 GHz processor? Find the CPI for the program.

(b) If the number of load instructions can be reduced by one-half, what is the

speed-up and CPI?

4. (12%) Suppose we have developed a new version of a processor with the following characteristics.

Version	Voltage	Clock rate
Version 1	5V	0.5GHz
Version 2	3.3V	1GHz

(a) By how much has the capacitive load been reduced between versions if the dynamic power has been reduced by 10%?

(b) By how much has the dynamic power been reduced if the capacitive load does not change?

(c) Assuming that the capacitive load of version 2 is 80% the capacitive load of version 1, find the voltage for version 2 if the dynamic power of version 2 is reduced by 40% from version 1.

5. (15%) In the following problems, the data table contains the values for registers `$t0` and `$t1`. You will be asked to perform several MIPS logical operations on these registers.

Case a.	`$t0 = 0x55555555, $t1 = 0x12345678`
Case b.	`$t0 = 0xBEADFEED, $t1 = 0xDEADFADE`

For the cases above, what are the values of `$t2` for the following sequence of instructions:

(a)
```
sll   $t2, $t0, 4
or    $t2, $t2, $t1
```

(b)
```
sll   $t2, $t0, 4
andi  $t2, $t2, -1
```

(c)
```
srl   $t2, $t0, 3
andi  $t2, $t2, 0xFFEF
```

6. (15%) Consider the following binary data showed in the table.

Case a.	$1010\ 1101\ 0001\ 0000\ 0000\ 0000\ 0000\ 0010_{two}$
Case b.	$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two}$

(a) Write the MIPS code that creates the 32-bit constants listed above and stores that value to register `$t1`.

(b) If the current value of the PC is `0x00000600`, can you use a single branch instruction to get to the PC address as shown in the table above? If the current value of the PC is `0x00400600`, can you use a single branch instruction to get to the PC address as shown in the table above?

(c) If the immediate field of a MIPS instruction was only 8 bits wide, write the MIPS code that creates the 32-bit constants listed above and stores that value to register `$t1`. **Do not use the `lui` instruction**.

7. (10%) Given a finite word length, overflow occurs when a result is too large to be represented accurately; however, underflow occurs when a number is too small to be represented correctly. The following table shows pairs of decimal numbers.

	A	B
Case a.	200	103
Case b.	247	237

(a) Assume A and B are signed 8-bit decimal integers stored in two's-complement format. Calculate A + B and A - B using saturating arithmetic. The result should be written in decimal. Show your work. Is there overflow, underflow, or neither?

(b) Assume A and B are unsigned 8-bit integers. Calculate A + B and A - B using saturating arithmetic. The result should be written in decimal. Show your work. Is there overflow, underflow, or neither?

8. (15%)

(a) What decimal number does the bit pattern represent if it is a floating-point number? Use the IEEE 754 standard.

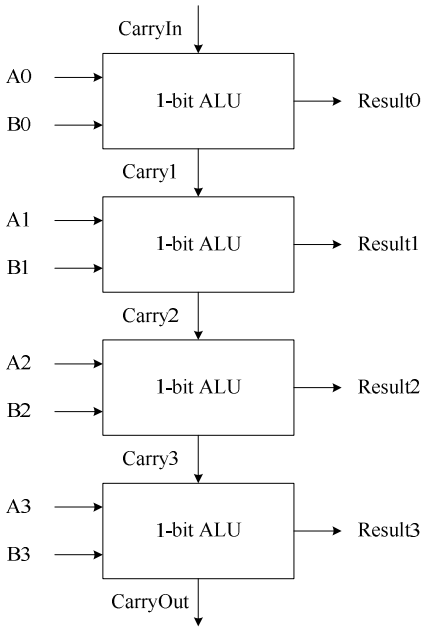
a.	`0x24A60004`
b.	`0xAFBF0000`

(b) Write down the binary representation of the decimal number `-938.8125`, assuming the IEEE 754 single precision format.

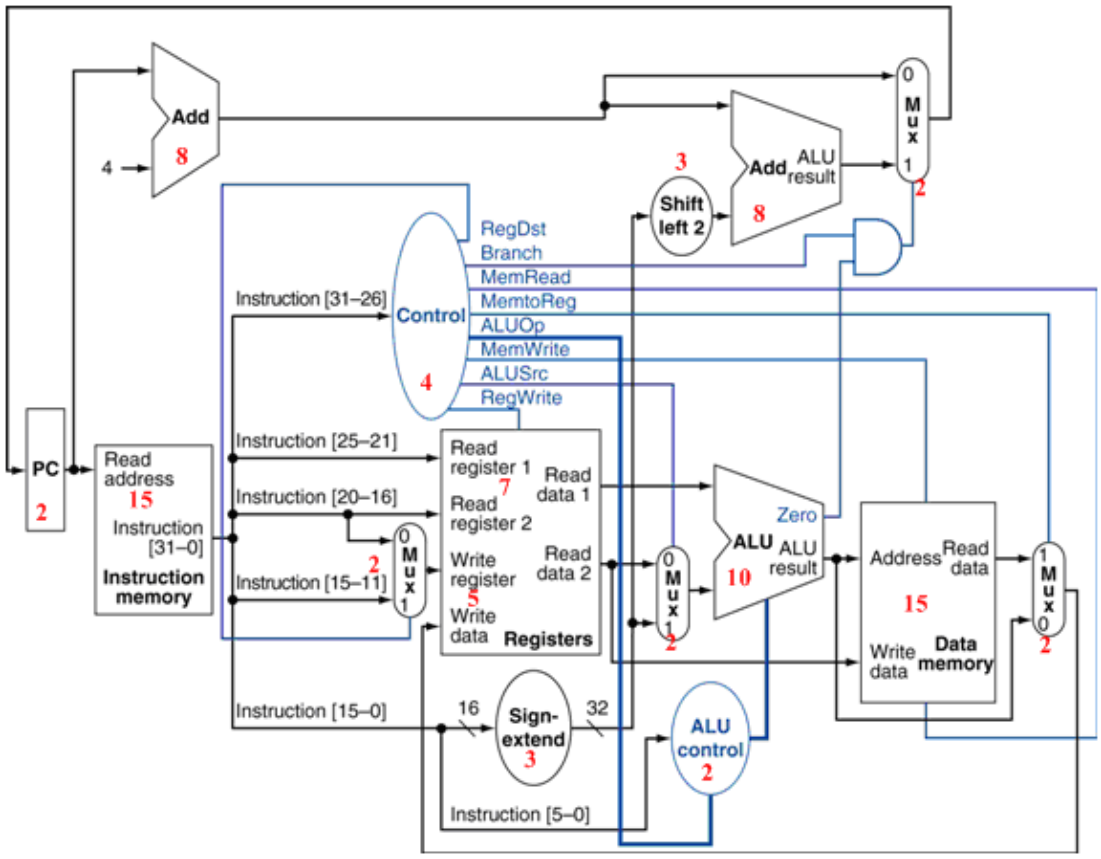
(c) Write down the binary representation of the decimal number `-1609.5`,

assuming the IEEE 754 double precision format.

9. (10%) 下圖為一4-bit ALU，請以最簡單的電路為此ALU偵測overflow，並解釋其原理



Good Luck !!



R-type: ADD R1, R2, R3
Load: LW R1, -12(R2)
Store: SW R1, -12(R2)

1. (a)

Type	Hardware elements used by instruction						
R-type	PC	IM(R)	REG(R)	MUX	ALU	MUX	REG(W)
Load	PC	IM(R)	REG(R)	ALU	DM(R)	MUX	REG(W)
Store	PC	IM(R)	REG(R)	ALU	DM(W)		

1. (b)

Type	Latencies (ns)							Total (ns)
R-type	2	15	7	2	10	2	5	43
Load	2	15	7	10	15	2	5	56
Store	2	15	7	10	15			49

1. (c)

Clock time: 56 ns, frequency 17.86 MHz

2.

Pass argument: $a0 ~ $a3
Return value: $v0, $v1

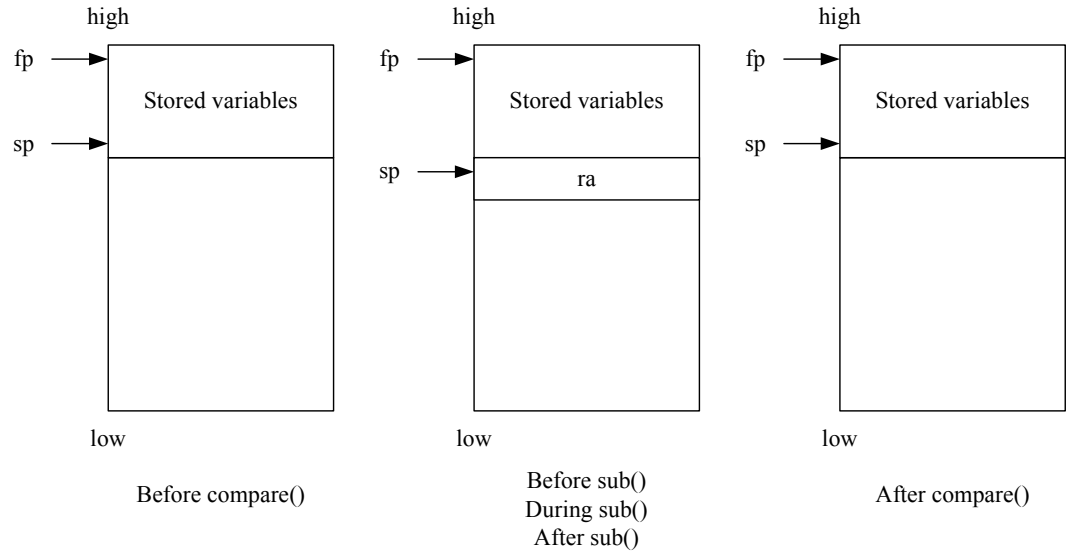
MIPS code:

```
compare:
        addi $sp, $sp, -4
        sw   $ra, 0($sp)

        add  $s0, $a0, $0
        add  $s1, $a1, $0

        jal  sub
        addi $t1, $0, 1
        beq  $v0, $0, exit
        slt  $t2, $0, $v0
        bne  $t2, $0, exit
        addi $t1, $0, $0
exit:
        add  $v0, $t1, $0
        lw   $ra, 0($sp)
        addi $sp, $sp, 4
        jr   $ra

sub:
        sub  $v0, $a0, $a1
        jr   $ra
```



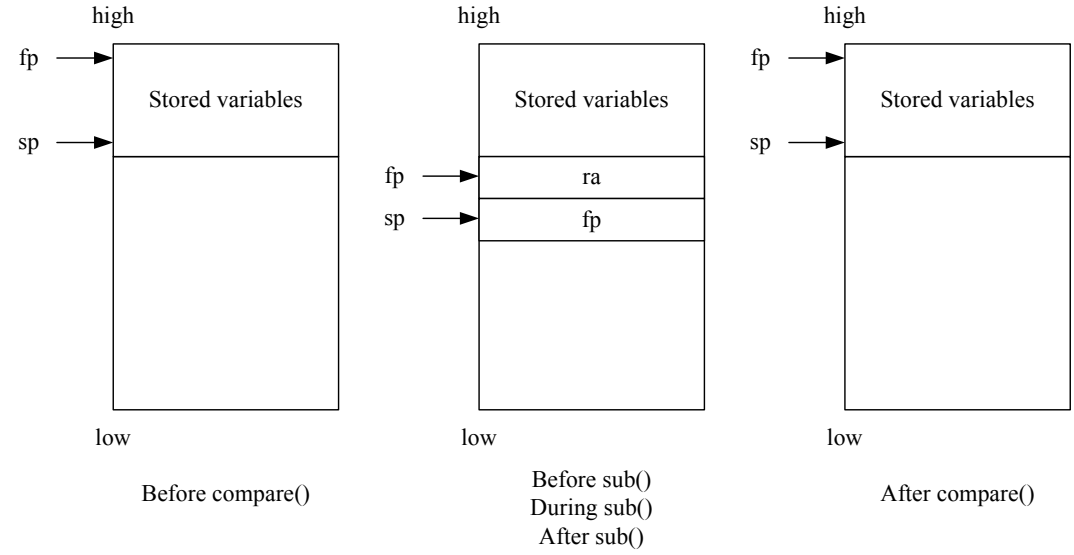
另一種解： MIPS code

```
compare:
        addi    $sp, $sp, -8
        sw      $fp, 0($sp)
        sw      $ra, 4($sp)

        add     $s0, $a0, $0
        add     $s1, $a1, $0

        jal     sub
        addi    $t1, $0, 1
        beq     $v0, $0, exit
        slt     $t2, $0, $v0
        bne     $t2, $0, exit
        addi    $t1, $0, $0
exit:
        add     $v0, $t1, $0
        lw      $fp, 0($sp)
        lw      $ra, 4($sp)
        addi    $sp, $sp, 8
        jr      $ra

sub:
        sub     $v0, $a0, $a1
        jr      $ra
```



3. (a)

Total cycles: $4 \cdot 500 + 4 \cdot 50 + 5 \cdot 100 + 3 \cdot 50 = 2850$

Total time = 2850 / 2 (ns) = 1425 ns = 1.425 us

CPI = 2850 / 700 = 4.071

3. (b)

Total cycles: $4 \cdot 500 + 4 \cdot 50 + 5 \cdot 50 + 3 \cdot 50 = 2600$
CPI = 2600 / 650 = 4
Speed-up = 2850 / 2600 = 1.096

4 (a)

$P1 = C_1 \cdot 25 \cdot 0.5 = 12.5\ C_1$
$P2 = C2 \cdot 10.89 \cdot 1 = 10.89\ C_2$
$P2 = 0.9 \cdot P1$
C2 / C1 = 1.033 (C2 爲原本的 1.033 倍，或 C2 增加 3.3%)

4 (b)

P1 = 12.5 C
P2 = 10.89 C
P2 / P1 = 0.872 (power 爲原本的 0.872 倍，或減少 12.8%)

4 (c)

$P1 = C \cdot 25 \cdot 0.5 = 12.5 \cdot C$
$P2 = 0.8\ C \cdot V^2 \cdot 1 = 0.8\ C \cdot V^2$
P2 = 0.6 P1 => V = 3.06 (V)

5 (a)

a. 0x57755778
b. 0xFEFFFEDE

5(b)

a. 0x00005550
b. 0x0000EED0

5(c)

a. 0x0000AAAA
b. 0x0000BFCD

6. (a)

Case a:

```
lui $t1, 0xad10
ori $t1, 0x0002
```

Case b:

```
lui $t1, 0xffff
ori $t1, 0xffff
```

6. (b)

PC + 4 + 0x1FFFC = 0x00020600
PC + 4 – 0x20000 = 0xFFFE0604

Case a: out of the range above (No!)
Case b: In the branch range (Yes!)

6. (c) 可以有不同的寫法, ex.

Case a:

```
sub     $t1, $t1, $t1
ori     $t1, 0xad
sll     $t1, 8
ori     $t1, 0x10
sll     $t1, 16
ori     $t1, 0x02
```

case b:

```
sub     $t1, $t1, $t1
subi    $t1, 1
```

7 (a)

Case a:

A = 200 = 0xC8(hex) (negative: -56 Dec)
B = 103 = 0x67(hex) (positive: 103 Dec)
A + B = -56 + 103 = 47
A – B = -56 – 103 = -159 (underflow, set to -128)

Case b:

A = 247 = 0xF7(hex) (negative: -9 Dec)
B = 237 = 0xED(hex) (negative: -19 Dec)
A + B = -9 – 19 = -28 (neither)
A – B = -9 + 19 = 10 (neither)

7 (b)

Case a:

A + B = 200 + 103 = 303 (overflow, set to 255)
A – B = 200 – 103 = 97 (neither)

Case b:

A + B = 247 + 237 = 484 (overflow, set to 255)
A – B = 247 - 237 = 10 (neither)

8 (a)

a. $1.2968754768 \times 2^{-54}$ (或 7.199×10^{-17})
b. $-1.4921875 \times 2^{-32}$ (或 -3.474×10^{-10})

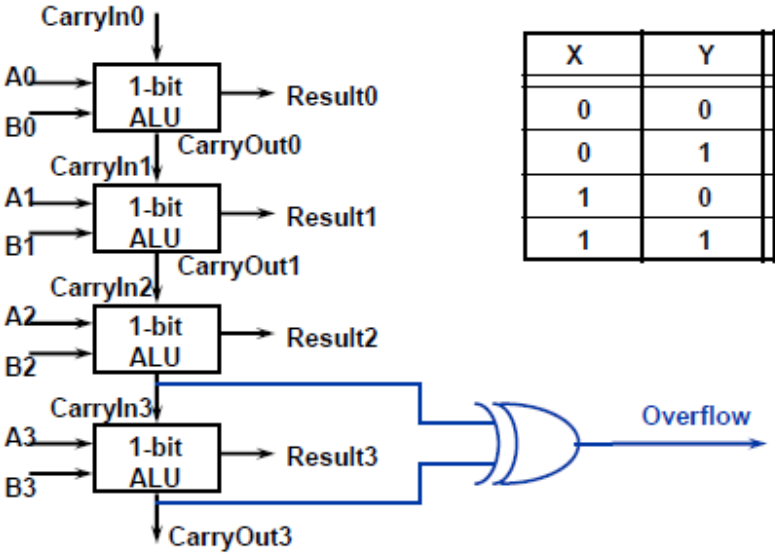
8(b)

-938.8125 = 1_10001000_11010101011010000000000

-1609.5 = 1_10000001001_10010010011000000000000000000000000000000000

9

Overflow = CarryIn[N-1] XOR CarryOut[N-1]



X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0