CMPE324 Midterm, 2017-2018/Spring
Date: 21/04/2018 **Duration: 100** min.

Number: …… **Name :** ……

Q1) [10pts] For a given program, two different compilers (Px, Py) were used for compilation on the same system. Given that Px generates programs that require 5% fewer instructions and have a 10% lower CPI than Py. Compute how fast Px as compared with Py. [you may calculate the CPU time for each case]

……

……

……

Q2) [15pts] The MIPS function given below receives one non-negative argument in $a0 and returns one output in $v0. What will this function return if it is called with an argument ($a0) of 4? Show the work in the given table.

```
func:li $t0, 1
     li $t1, 1
loop:blti $a0, 2, exit
     add $t2, $t0, $t1
     move $t0, $t1
     move $t1, $t2
     addi $a0, $a0, -1
     j loop
exit:move $v0, $t1
     jr $ra
```

$a0	$t0	$t1	$t2	$v0

Describe in English what the function actually computes?

……

Q3) [15pts]

For each of the following high-level code segments, complete the given equivalent MIPS assembly implementation.

C++ Code segment	**Equivalent MIPS**
int pow = 1; int x = 0; while (pow != 256) { pow = pow * 4; x = x + 1; }	# $s0 = pow, $s1 = x addi $s0, $0, 1 add $s1, $0, $0 addi $t0, $0, 256 while: .... addi $s1, $s1, 1 ..... done:
int x; if (x ≥ 0 && x < 8) x++;	# $s1 = x addi $s0, $zero, 8 .... addi $s1, $s1, 1 exit:

Q4) [10pts] Consider the following binary representation of **π**. Based on the IEEE754 standard representation of floating point numbers; specify the corresponding sign, the exponent, and the fraction fields.

0.01100 1001 0000 111 1101 1011 X 2^3

Sign	**exponent**	**Fraction**

Q5) [15pts] Given the following C++ function:

```
float  CelToFahr(float Cel)
{
     return ((9.0/5.0)*Cel+32.0);
}
```

Using single precision floating point arithmetic, complete the following MIPS implementation. Let the value of Cel be in $f12 register and let the result be in $f0 register.

CelToFahr:

```
# Load $f2 with 32.0, $f4 with 9.0, $f6 with 5.0
.....
.....
.....
# Compute Fahr and let the result be in $f0
.....
.....
.....
j $ra
```

Q6) [20pts] Consider the following MIPS assembly code:

```
begin:   addi $t0, $zero, 0
         addi $t1, $zero, 1
loop:    slt $t2, $a0, $t1
         bne $t2, $zero, finish
         add $t0, $t0, $t1
         addi $t1, $t1, 2
         j loop
finish:  add $v0, $t0, $zero
```

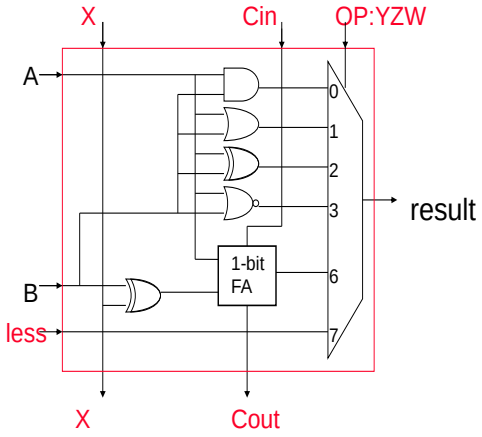
Fill in the table below the missing machine code. Also, write the equivalent high-level language code. Note the following MIPS instructions formats:

Inst. Type	31-26	25-21	20-16	15-11	10-6	5-0
R	op= 0	rs	rt	rd	shamt	func
I	op= 1, 4-62	rs	rt	Immediate		
J	op= 2, 3	target address				

Note: divide by 4 means shift logically right by two bits.

Address	**Machine code**	**C++ Code**
00FFF800	**001000 00000 01000 00000 00000 000000**	
00FFF804	**001000 00000 01001 00000 00000 000001**	
00FFF808	**000000 00100 01001 01010 00000 101010**	
00FFF80C	.....	
00FFF810	**000000 01000 01001 01000 00000 100000**	
00FFF814	**001000 01001 01001 00000 00000 000010**	
00FFF818	.....	
00FFF81C	.....	

Q7) For the given 1-bit ALU circuit, **(a) [7pts]** Fill in the given table.



X	Y	Z	W	Cin	Operation
X	0	0	0	x	
X	0	0	1	x	
X	0	1	0	x	
X	0	1	1	x	
0	1	1	0	0	
1	1	1	0	1	
1	1	1	1	1	

(b) [8pts] Use this 1-bit ALU to implement **slt (set less than)** instruction. Make the necessary connections and indicate the value of less input. Also, make the necessary connection for checking the **overflow** status.

$Xc_{in}YZW$=.....