**Q1.** Determine whether each of the following statements is True or False? <2 points>

1.5

a. ***Super computers*** rely on parallelism to achieve very high performance for high end applications. F

b. In PostPC era, the cloud server is ***only used for storage*** and all applications must be installed on the PMD. F

c. A CPU is designed with two adders, where only one adder is used for execution and the other one is used a spare in case the first adder becomes faulty. This is an example of improving ***performance via prediction***. F

d. Flash drive is faster and more expensive than the traditional magnetic disk drive. T

(Idavg & (Ic,) same

**Q2.** CPU-1 and CPU-2 have the ***same ISA and same compiler***. The ISA has three instruction types: A, B, and C. The CPI for each instruction type in CPU-1 and CPU-2 are given in the table below. ProgramX is executed on both CPUs. The relative frequency of each instruction type in ProgramX is also given in the table. Answer the following questions:

1.5 <3 points>

| | A | B | C |
|---|---|---|---|
| **CPU-1 CPI** | 2 | 3 | 2 |
| **CPU-2 CPI** | 4 | 1 | 1 |
| **ProgramX Instruction%** | 20% | 40% | 40% |

$\frac{IC_i}{IC}$

a. Compute ***$CPI_{avg}$*** for ProgramX when executed on CPU-1 and CPU-2.

$CPI_{avg}\ CPU_1 = 2 \times 0.2 + 3 \times 0.4 + 2 \times 0.4$

$CPI_{avg}\ CPU_2 = 4 \times 0.2 + 1 \times 0.4 + 1 \times 0.4$

| CPU-1 $CPI_{avg}$ = 2.4 | CPU-2 $CPI_{avg}$ = 1.6 |
|---|---|

b. Given that the Clock Rate of CPU-1 is ***3 times*** the Clock Rate of CPU-2, which CPU has bette performance when executing ProgramX? ***and by how much?***

$performance = \frac{1}{cpu\ time}$

**Q3.** ***Two compiled versions*** (Version-1 and Version-2) of the same program are executed on the ***same CPU***. The IC for each instruction type in Version-1 is given in the table. The CPI for each instruction type is given as well. Answer the following questions: 1.5 <3 points>

| Instruction Type | A | B | C | D |
|---|---|---|---|---|
| Version-1 IC | 3 | 6 | 2 | 2 |
| CPI | 4 | 1 | 5 | 2 |

a. How many ***clock cycles*** are needed for Version-1?

clk cycles = IC x CPI = 3x4 + 6x1 + 2x5 + 2x2

Ic = 13

Version-1 Clock Cycles = 32 ✓

b. Given that the performance of Version-2 is ***4 times*** the performance of Version-1, how many ***clock cycles*** are needed for Version-2?

$CPI_A = 4 \times \frac{3}{13} + 1 \times \frac{6}{13} + 5 \times \frac{2}{13} + 2 \times \frac{2}{13}$

$CPI_A = \frac{32}{13}$

$\frac{Perf\ 2}{Perf\ 1} = 4$

CPU time = Ic x T x CPI avg

CPU time = clk cycles x T

$\frac{CPU\ time_2}{CPU\ time_1} = \frac{clk\ cycle_2}{clk\ cycle_1} = \frac{1}{4}$ This is

Version-2 Clock Cycles = 1 ✗

**Q4.** Answer the following short questions:

a. Convert the following C-statement into RISC-V assembly language. ***Assume that "B" is an array of long long integers and its base address is mapped to register x8***: <1 point> 0

B[5] = 0; 5x8

RISC-V Assembly: ld $X_0$, 40($X_8$)

b. Convert the following C-statement into RISC-V assembly language. ***Assume that variable W is mapped to register x20 and variable Y is mapped to register x10***: <1 point> 0.5

$X_{20}$ $X_{10}$

$W = \bar{Y}$;

0xFFF

RISC-V Assembly: Xori $X_{20}$, $X_{10}$, -1 ✓

c. Convert the following RISC-V machine code instruction into assembly language. A list of opcodes and function fields is given on the right. <2 points> 1.5

Machine code instruction: 00000000111100111101001010010011

$X_9$ $X_5$

| Instruction | Opcode | Funct3 | Funct6 or Funct7 |
|---|---|---|---|
| add | 0110011 | 000 | 0000000 |
| lh | 0000011 | 001 | n.a. |
| bge | 1100111 | 101 | n.a. |
| addi | 0010011 | 000 | n.a. |
| slli | 0010011 | 001 | 000000 |
| srli | 0010011 | 101 | 000000 |

RISC-V Assembly: srli $X_5$, $X_{15}$, 15 ✓ ✓ ✓

$X_7$

d. Specify the contents of register x6 and memory location at address 2 (i.e. Memory[2]) in ***hexadecimal*** format after executing the following RISC-V code. ***Assume the initial values are as follows:*** ***x20 = 0x C7B2345D8A09FEB1, Memory[2] = 0xAD, Memory[3] = 0x74***: *<2 points>*

sh x20, 2 (x0)

lb x6, 3 (x0)

(x6) = ---------0x74---------

Memory[2] = ~~0xC7B2345D8~~

e. Specify the contents of registers x17 and x18 in ***hexadecimal*** format after executing the following RISC-V code: *<2 points>*

lui x17, 0x7A2D3

addi x18, x17, 0xC59

(x17) = ---------0x7A---------

(x18) = ---------0x7AC59---------

f. Given the following RISC-V code: *<2 points>*

| PC in decimal | Instruction |
|---|---|
| 20 | Loop: beq x6, x0, 8 |
| 24 | add x5, x5, x6 |
| 28 | addi x6, x6, -1 |
| 32 | jal x0, Loop |
| 36 | add x7, x5, x0 |

- If we are currently executing the branch instruction at ***PC = 20*** and ***(x6) = 0x0000000000000000***, which instruction will be executed next?

Answer: ---------36---------

- Compute the value of the "Loop" immediate in the "jal" instruction at ***PC = 32***.

Loop (in ***decimal*** format) = ---------−6---------

e. Given that the code on the right uses lock/unlock synchronization and two processes (P1 and P2) are executing the code in parallel, answer the following questions: *<2 points>*

```
try:  addi x5,x0,1
      lr.d x6, (x30)
      bne x6,x0,try
      sc.d x8,(x30),x5
      bne x8,x0,try
      ld x21, 40(x0)
→     addi x21, x21, 1
      sd x21, 40(x0)
      sd x0, 0(x30)
```

- If ***P1*** is currently executing the addi x21, x21, 1 instruction as indicated by the black arrow, determine the value of the Lock variable in the memory and determine which instruction ***P2*** will probably execute next?

Lock variable = ---------

P2 will probably execute next (list all possible instructions): ---------

---

- Which instruction represents the 1st instruction in the ME region?

1st instruction in ME region is: ---------bne x6, x0, try---------

**Q5.** Given the C-language procedure on the right, answer the two questions below. Assume that the base address of "array A" is mapped to x10, , variable "n" is mapped to x11, variable "key" is mapped to x12, variable "result" is mapped to x13, and variable "i" is mapped to x27. ***Notice that x27 is a saved register.***

*<6.5 points>* 3

a. Translate the procedure into RISC-V assembly language.

```
long long int REC (long long int A[],
long long int n, long long int key)
{
        int i, result;
        result = 0;
        for (i = 0; i < n; i++)
                if (A[i] == key)
                        result ++;
        return result;
}
```

```
addi  SP , SP , -40
sd X27 , 0(SP)
sd X13 , 8(SP)
sd X1 , 16(SP)
sd X11 , 24(SP)
addi X13 , X0 , 0
addi X27 , X0 , 0
bge X27 , X11 , Exit
slli X10 , X27 , 3
beq X10 , X12 , L1
beq X0 , X0 , Exit
L1 : addi X13 , X13 , 1
ld X27 , 0(SP)
ld X13 , 8(SP)
ld X1 , 16(SP)
ld X11 , 24(SP)
ld X12 , 32(SP)
addi SP , SP , +40
jalr X0 , 0(X1)
Exit : addi SP , SP , +40
jalr X0 , 0(X1)
```

0.5 0.5 0.5 0.25 X 1

b. Given that register ***x11 is initialized to 100*** and register ***x12 is initialized to 17***, translate the following C-statement associated with the procedure above into RISC-V assembly language. ***Assume variable Number is mapped to register x9 and the starting address of "array A" in memory is 16.***

Number = REC (address of array A, 100, 17);

RISC-V Assembly:

```
addi X10 , X0 , 16   X
addi X11 , X0 , 100
addi X12 , X0 , 17   No need
```

**Q6.** Given the following ***Non-leaf*** procedure written in RISC-V assembly language, answer the questions below accordingly. ***The procedure has two arguments mapped to registers x10 and x11 and the return value is mapped to register x12.*** *<3.5 points>*

```
PW:
        Push to Stack
        bne x11, x0, Else
        addi x12, x0, 1
        Pop from Stack (1)
        jalr x0, 0(x1)
Else:
        addi x11, x11, -1
        jal x1, PW
        mul x12, x10, x12
        Pop from Stack (2)
        jalr x0, 0(x1)
```

a. Determine whether each of the following statements is True or False?

- Register x1 ***must*** be saved in the stack
- Register x10 ***must*** be saved in the stack
- Register x11 ***must*** be saved in the stack

b. Given the following procedure call, what is the ***final value*** in register x1 and x12 in ***decimal*** format?

***Procedure call:***

| PC in decimal | Instruction |
|---|---|
| 40 | addi x10, x0, 4 |
| 44 | addi x11, x0, 2 |
| 48 | jal x1, PW |

(x1) = 48 + 4 = 52

(x12) = 1