Computer Architecture
Quiz 1

October 8, 2024

Student id: Name:

---

1. (30%) Question 1 Assume that, for arithmetic, load/store, and branch instructions, a processor has CPIs of 3, 15, and 10, respectively. If there is only one processor, a program requires executing 5×10^9 arithmetic instructions, 1×10^9 load/store instructions, and 500×10^6 branch instructions. Each processor has a 2 GHz clock frequency. Also, assume that, as the program is parallelized to run over multiple cores, the number of arithmetic and load/store instructions per processor is divided by $0.75 \times p$ (where p is the number of processors). However, the number of branch instructions remains the same.
   (a) (10%) Find the total execution time (in seconds) for this program on 1, 2, 4, and 8 processors.
   (b) (10%) **To what** should the CPI of load/store instructions be reduced for a single processor to match the performance of **2** processors using original CPI values?
   (c) (10%) **To what** should the clock frequency be increased for a single processor using original CPI values to match the performance of **8** processors using original CPI values? (Represent your answer in GHz)

(1)answer: 17.5/12.5/7.5/5(s)
-3 points per wrong answer, points will be deducted until all 10 points are used up.
(2)answer: 5
-5 points for wrong answer, -5 points for answer error but correct formula
(3)answer: 7(GHz)
-5 points for wrong answer, -5 points for answer error but correct formula, -2 for wrong unit

2. (35%) Question 2 Consider the following RISC-V assembly code:

```
addi x18 x0 1
addi x19 x0 8
addi x20 x0 9

LOOP:
    mul x30 x18 x19
    add x30 x30 x21
    ld x5  -8(x30)
    ld x6   8(x30)
    ld x7   0(x30)
    add x10 x5 x6
    add x10 x10 x7

    sd x10, 0(x22)
    addi x22, x22, 8
    addi x18, x18, 1
    blt x18, x20, LOOP
```

Assume that register `x21` holds the base address of the array `D` and register `x22` holds the base address of the array `Answer`. The value in array `D` is as the following:

index	0	1	2	3	4	5	6	7	8	9
value	1	2	3	4	5	6	7	8	9	10

NOTE: This is a 64-bit system.

(a) (15%) What are the results of the array `Answer`? (List all the values of `Answer[i]` that have been written by operation `sd`)

(b) (10%) For the RISC-V assembly code above, how many RISC-V instructions are executed?

(c) (10%) What is the function of the code above?

ANS:

(a)

index	0	1	2	3	4	5	6	7
value	6	9	12	15	18	21	24	27

-2 points per wrong Answer[i] ($i \notin [0, 7]$ or wrong Answer[i] values)

(b) $11 \times 8 + 3 = 91$

1. -10 points if the answer is wrong
2. -3 if the calculating process is correct but answer is wrong.

(c) $Answer[i-1] = D[i-1] + D[i] + D[i+1]$ for index i starting from 1 to 8 in array D.

1. id convolution (5%), window size = 1*3 (5%)
2. If you describe the code correctly by text, you will also get 10 points.
3. If your (a) is correct but there are some errors or unclear description in your answer, you will also get 10 points.
4. If you provide the correct Python or C or C++ code, you will also get 10 points.
5. If your (a) is correct and there is minor error in your code (such as index), you will still get 10 points.
6. If your code and the answer in (a) is not consistent, you will get 0 point.
7. If your (a) is not correct and there are errors in your (c), -3 points each error.

3. (35%) Question 3

A zero-address machine is a stack-based machine where all operations are done using values stored on the operand stack. For this problem, you may assume that its ISA allows the following operations:

> **PUSH M** - pushes the value stored at memory location M onto the operand stack.
>
> **POP M** - pops the operand stack and stores the value into memory location M.
>
> **OP** - Pops two values off the operand stack, performs the binary operation OP on the two values, and pushes the result back onto the operand stack. The popped values are **NOT** stored back to the memory. (hint: To compute A - B with a stack machine, the following sequence of operations is necessary: PUSH A, PUSH B, SUB. After execution of SUB, A and B would no longer be on the stack, but the value A-B would be at the top of the stack.)

A three-address load-store machine's sources and destination are registers. Values are loaded into registers using memory operations (The RISC-V is an example of a three-address load-store machine). For this problem, you may assume that its ISA allows the following operations:

> **OP R3, R1, R2** - Performs a binary operation OP on the values stored at registers R1 and R2 and stores the result back into register R3 (R3 = R1 OP R2).
>
> **LOAD R, M** - Loads the value at memory address M into register R.
>
> **STORE R, M** - Stores the value in register R into memory address M.

To measure memory efficiency, we make the following assumptions about the instruction sets:

> The opcode is always 1 byte.
>
> All register operands are 1 byte.
>
> All memory addresses are 2 bytes.
>
> All data values are 4 bytes.

There are no other optimizations to reduce memory traffic, and the variables A, B, and C are initially in memory.

You are only allowed to use the following instructions: For zero-address machine: PUSH, POP, ADD, and SUB. For three-address load-store: LOAD, STORE, ADD, and SUB.

(a) (15%) Write the code sequences for the following high-level language fragment for each architecture style. **Be sure to store the contents of A and C back into memory, but do not modify any other values in memory. You should use the least operations to finish the code.**
A = A + B;
C = A - C;

(b) (10%) Which architecture is more efficient as measured by the total number of bytes of transferred instructions. **You should write the number of bytes on the answer sheet.**

(c) (10%) Which architecture is more efficient as measured by the total number of bytes of transferred data value. **You should write the number of bytes on the answer sheet.**

Hint: Take operation **PUSH M** of zero-address machine for example: to transfer the instruction, it needs 3 bytes(1 for opcode, 2 for memory address), and to transfer the data value, it needs 4 bytes.

ANS:

(a)

Not least operations (-3 for each architecture)
Wrong content (-4 for each content)
Wrong function (-7.5 for each architecture)

ISA	Opcode	Operands	Instruction bytes	Data bytes
Zero-address machine	PUSH	A	3	4
	PUSH	B	3	4
	ADD		1	0
	POP	A	3	4
	PUSH	A	3	4
	PUSH	C	3	4
	SUB		1	0
	POP	C	3	4
			20	24
Three-address load-store machine	LOAD	R1, A	4	4
	LOAD	R2, B	4	4
	ADD	R1, R1, R2	4	0
	STORE	R1, A	4	4
	LOAD	R3, C	4	4
	SUB	R3, R1, R3	4	0
	STORE	R3, C	4	4
			28	20

(b)
For each calculation error (-3)
Wrong summary (-4)

Zero-address machine is more efficient. ($20 < 28$)
(c)
For each calculation error (-3)
Wrong summary (-4)

Three-address load-store machine is more efficient. ($20 < 24$)