

Computer Organization and microprocessor – ENCS2380

**Midterm – First semester 2022/2023**

Name (Arabic): ID:

|  |  |  |  |
| --- | --- | --- | --- |
| **Question** | **Maximum** | **Mark** | **ABET SO** |
| **1** | 25 |  | A |
| **2** | 30 |  | C |
| **3** | 20 |  | A |
| **Total** | **75** |  |  |

**Question 1: [25 marks]**

**A)** A compiler designer is trying to decide between two code sequences for a particular CPU. The hardware designers have supplied the following facts:

|  |  |
| --- | --- |
| **Instruction class** | **CPI** |
| A | 1 |
| B | 3 |
| C | 4 |

For a particular high-level language, the compiler writer is considering generating two code sequences that require the following instruction counts:

|  |  |  |  |
| --- | --- | --- | --- |
| Code Sequence | Instruction Count  [in millions] | | |
| **A** | **B** | **C** |
| **1** | 2 | 1 | 2 |
| **2** | 4 | 3 | 1 |

1- What is the average CPI for each sequence? [10pts]

Avg CPI (1) = 1x 2/5 + 3x1/5 + 4 x2/5 = 13/5 = **2.6**

Avg CPI (2) = 1x4/8 + 3x3/8 + 4x1/8 = 17/8 = **2.125**

2-Which code sequence is faster and by how much? [5pts]

CPU time (1) = IC (1) \* Avg CPI (1) \* Clock cycle

CPU time (2) = IC (2) \* Avg. CPI (2) \* Clock cycle

Clock cycles (1) = 5 million \* 2.6 = **13 million clock cycles**

Clock cycles (2) = 8 million \* 2.125 = 17 million clock cycles

* **Code sequence (1) is faster by 17/13 = 1.3**

**B)** In the protein string matching code, the current machine needs 4 days of execution time. If 20% of the time, the processor is doing integer instructions, and 35% of the time doing I/O operations, which is better solution to improve the performance:[clarify your answer] [10pts]

**Solution1**- Compiler optimization that reduces number of integer instructions by 25% (assume each integer inst. takes the same amount of time)

**Solution2**- Hardware optimization that reduces the latency of each IO operations from 6us to 5us.

Sol1: f=0.2, s=1.25

Overall = 1/(0.8 + 0.2/1.25) = **1.0416**

Sol2: f=0.35, s as percentage = 1/6 = 16.67%

Overall = 1/(0.65 + 0.35/1.1667) = **1.052**

Therefore, Solution 2 is better

**Question 2: [30 marks]**

|  |  |
| --- | --- |
| **Main Memory** | |
| Address | Content |
| *0* | *0x8C06* |
| *1* | *0x6C07* |
| *2* | *0x8014* |
| *3* | *0x2408* |
| *4* | *0xA000* |
| *… …* | *… …* |
| *6* | *0x0004* |
| *7* | *0xFFFF* |
| *8* | *0x0002* |
| *…. …* | *…. …* |

A computer has 16-bit instructions with instruction format consists of 4 bits opcode, 1 bit for register, 1 bit (M) that indicate the addressing mode, and the remaining bits are used either for immediate or address. When M=0 => immediate, when M=1 => memory direct. The memory consists of 16-bit cells. Assume that a programs (instructions) is stored in the memory starting from address 0 to 4, while data which is 2’s complement is stored starting from address 6 to 8. Unless otherwise stated, assume that the initial content of all registers and memory is 0. All numbers begin with 0x are in hexadecimal.

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode (4bits) | Reg(1bit) | M (1bit) | Immediate/Memory Address |

Instruction format

The computer supports the following operations: **0110**: Add, **1000**: Load, **0010**: Subtract, **0100**: Store, **1010**: Jump

1000 1 1 000000 0110

Load R1, [6]

0110 1 1 0000000111

ADD R1,[7]

1. Starting with zero’s initial values in all registers, fill in the following table by executing four instructions. [10pts]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instruction** | **PC** | **IR** | **MAR** | **MBR** | **R1** |
| Initial values | 0 | 0 | 0 | 0 | 0 |
| After executing first instruction at address 0 | 1 | 0x8C06 | 6 | 0x0004 | 0x0004 |
| After executing second instruction at addresss 1 | 2 | 0x6C07 | 7 | 0xFFFF | 0x0003 |

2. What is the maximum memory size that can be accessed directly by this machine? [6pts]

# of cells = 2^10 x 2 Bytes = **2KB**

3.Re-write the stored program at address 0 to 4 in assembly code. [5pts]

Load R1, [6]

Add R1, [7]

Load R0, 20

Sub R0, [8]

Jump 0

4. Write the RTL micro operations for the instruction at address 4.[4pts]

MAR 🡨 PC

MBR 🡨 M[MAR]

IR 🡨 MBR

PC 🡨 IR(9:0)

5. What is the range of the signed integer values that can be stored in one memory cell? [5pts]

Range: -2^15 🡪 + 2^15 -1

**Question 3: [20 marks]**

For each instruction below, answer the following questions:

A. What is the addressing mode of the instruction? We consider these modes: Immediate, memory direct, memory indirect, register direct, register indirect, register indirect with displacement, and register indexed.

B. What is the effective address of each instruction?

C. What will be the value of register **R0** after the instruction is executed?

Before each instruction, the initial content of memory and registers is shown in the figure below. Please note that, the instructions are executed individually. The execution of each instruction doesn’t depend on the results of other instructions.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Registers** |  | | 0xFFFFFFFF | R0 | | 0x00001010 | R1 | | 0x00001004 | R2 | | 0x00001008 | R3 | | 0x00000010 | R4 | | |  |  | | --- | --- | | **Address** | **Memory Content** | | 0x00001000 | 0x00001011 | | 0x00001004 | 0x00001020 | | 0x00001008 | 0x00001033 | | 0x0000100C | 0x00001044 | | 0x00001010 | 0x00001055 | | 0x00001014 | 0x00001066 | | 0x00001018 | 0x00001077 | | 0x0000101C | 0x00001088 | | 0x00001020 | 0x00001099 | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction** | **Addressing mode** | **Effective Address (EA)** | **Value of R0 after execution** |
| Load R0, [0x0001010] | Memory direct | 0x00001010 | 0x00001055 |
| Load R0, [R1 - 4] | Reg. indirect with disp. | 0x0000 100C | 0x00001044 |
| Load R0, [[0x00001004]] | Mem. Indirect | 0x0000 1020 | 0x00001099 |
| Load R0, [R1 + R4] | Reg. Indexed | 0x0000 1020 | 0x00001099 |
| Load R0, 0x00001018 | Immediate | No EA | 0x00001018 |