Midterm Exam CPE 517 WS-1

Matthew Raghunandan - 10460030

(I hope you don't mind that I typed this, my handwriting is absolutely awful and I also didn't realize I needed a pen and paper before opening the exam. I tried my best to format everything in so it's easy to follow what I did.)

Question 1

a

5 components of a computer:

1. Input
- Keyboard, Mouse
2. Storage
- Hard Drive, SSD
3. Network
- Ethernet, Wi-Fi
4. Processing
- CPU, GPU
5. Output
- Monitor, Speakers

b

Hexadecimal Calculations: FC + AE

FC

$$(15 \times 16^1) + (12 \times 16^0) = 252$$

AE

$$(10 \times 16^1) + (14 \times 16^0) = 174$$

FC + AE

$$252 + 174 = 426$$

426 in hex

$$426/16 = 26 \text{ remainder } 10$$
$$26/16 = 1 \text{ remainder } 10$$

1/16 = 0 remainder 1

426 in hex is **1AA**

Question 2

```
  00110011
- 00010000
----------
  00100011
```

Question 3

a

$$X9 = 0 + 8 = 8$$
$$X1 = X9 * 2 * 2 = 32$$
$$X1 = 32 + 1 = 33$$

So **X9** is 8 and **X1** is 33.

b

```
// Initialize i to 1
ADDI X2, XZR, #1

// Loop
Loop:
    // Check if i <= a
    SUBS XZR, X2, X19
    B.GT End

    // d[i] address = X5
    LSL X5, X2, #3
    ADD X5, X22, X5

    // d[b] address = X6
    LSL X6, X20, #3
    ADD X6, X22, X6

    // d[3] address = X7
    ADDI X7, X22, #24

    // d[b] value = X8
    LDUR X8, [X6, #0]

    // d[3] value = X9
    LDUR X9, [X7, #0]

    // d[i] = d[b] + d[3]
    ADD X8, X8, X9

    // Store d[i] value
    STUR X8, [X5, #0]

    // Increment i
    ADDI X2, X2, #1

    // To the beginning of the loop
    B Loop

End:
```

Question 4

a

An incredibly low-leveled programming language that essentially is a human-readable version of machine code. Instead of using binary, assembly language uses text to represent each instruction. It is how the computer understands what to do.

b

Machine code is the actual binary code that the computer processes. This is the lowest level you can go as each bit is specifically designed to be sent to the processor to perform a specific task.

c

Amdahl's Law tells us how much a program can be sped up by changing a signle part of it. It takes in account the amount of time that the part of the program is actually used.

Execution Time After Improvement = (Time Affected by Improvement / Improvement Factor) + Time Unaffected

Question 5

a

P1

$$1.5 \text{ CPI} = \frac{1}{1.5} = 0.6667 \text{ instructions per cycle}$$
$$3.5 \times 10^9 \text{ cycles per second} \times 0.6667 \text{ instructions per cycle} = 2.333 \times 10^9 \text{ instructions per second}$$

P2

$$1 \text{ CPI} = \frac{1}{1} = 1 \text{ instructions per cycle}$$
$$3 \times 10^9 \text{ cycles per second} \times 1 \text{ instructions per cycle} = 3 \times 10^9 \text{ instructions per second}$$

P3

$$2.5 \text{ CPI} = \frac{1}{2.5} = 0.4 \text{ instructions per cycle}$$
$$4.5 \times 10^9 \text{ cycles per second} \times 0.4 \text{ instructions per cycle} = 1.8 \times 10^9 \text{ instructions per second}$$

P2 is the best in terms of instruction per second.

b

P1

$$3.5 \times 10^9 \text{ cycles per second} \times 10 \text{ seconds} = 35 \times 10^9 \text{ cycles}$$
$$35 \times 10^9 \text{ cycles} \times 0.6667 \text{ instructions per cycle} = 23.333 \times 10^9 \text{ instructions}$$

P2

$$3 \times 10^9 \text{ cycles per second} \times 10 \text{ seconds} = 30 \times 10^9 \text{ cycles}$$
$$30 \times 10^9 \text{ cycles} \times 1 \text{ instructions per cycle} = 30 \times 10^9 \text{ instructions}$$

P3

$$4.5 \times 10^9 \text{ cycles per second} \times 10 \text{ seconds} = 45 \times 10^9 \text{ cycles}$$
$$45 \times 10^9 \text{ cycles} \times 0.4 \text{ instructions per cycle} = 18 \times 10^9 \text{ instructions}$$

Question 6

```
// Initialize prev_number to 0 and number to 1
SUBS X1, XZR, XZR
ADDI X2, XZR, #1

// Initialize i to 0
SUBS X3, XZR, XZR

for:
    // Check if i < Range
    SUBS XZR, X3, X19
    B.GE End

    // next_num = prev_number + number
    ADD X4, X1, X2

    // Store next_num in the array
    LSL X5, X3, #3
    ADD X5, X20, X5
    STUR X4, [X5, #0]

    // prev_number = number
    ADDI X1, X2, #0

    // number = next_num
    ADDI X2, X4, #0

    // Increment i
    ADDI X3, X3, #1

    // Jump to the beginning of the loop
    B for

End:
```