**Q1**

**1. Boolean Expression of the Circuit**

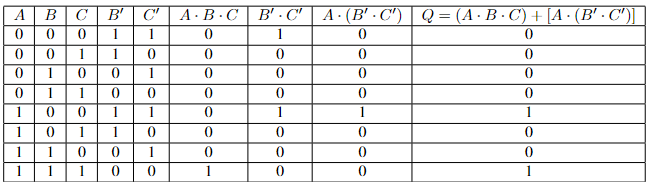
The Boolean expression for the circuit is:

***Q = (A⋅ B ⋅ C) + [A⋅ (B′⋅ C′)]***

Where:

* ***A⋅ B ⋅ C*** represents the output of the first AND gate.
* **B′** represents the NOT of **B**.
* **C′** represents the NOT of **C**.
* **B′ ⋅ C′** represents the output of the second AND gate after the NOT gates.
* **A⋅ (B′ ⋅ C′)** represents the output of the third AND gate.
* The final output **Q** is the OR of the two AND gates' outputs.

**2. Truth Table**



**Q2**

**A. Represent the two decimal numbers in One’s Complement and Two’s Complement representation using 8-bit binary:**

**i. Represent 33 and -121 in One’s Complement using 8-bit binary:**

* **33**:
  + Convert to binary: 33 in binary is 00100001.
  + Since it’s positive, One’s Complement remains the same: 00100001.
* **-121**:
  + Convert to binary: 121 in binary is 01111001.
  + Invert all the bits to get the One’s Complement: 10000110.

**ii. Represent 33 and -121 in Two’s Complement using 8-bit binary:**

* **33**:
  + Convert to binary: 33 in binary is 00100001.
  + Since it’s positive, Two’s Complement remains the same: 00100001.
* **-121**:
  + Start with the One’s Complement: 10000110.
  + Add 1 to the One’s Complement: 10000110 + 1 = 10000111.
  + Two’s Complement of -121: 10000111.

**B. Convert the positive number Ν=1010000001101 in single precision floating point format:**

* **Step 1**: Normalize the binary number:
  + The given number in binary is 1010000001101.
  + Normalize it to form 1.010000001101 × 2^12.
* **Step 2**: Identify the components of the IEEE 754 single precision format:
  + Sign bit: Since the number is positive, the sign bit is 0.
  + Exponent: The exponent is 12 (the power of 2 in the normalized form). In IEEE 754, the exponent is biased by 127. So, 12 + 127 = 139, which in binary is 10001011.
  + Mantissa: The fractional part 010000001101 will be stored in the mantissa (23 bits, padded with zeros if necessary).
* **Step 3**: Combine them:
  + Sign bit: 0
  + Exponent: 10001011
  + Mantissa: 01000000110100000000000
* **Final Single Precision Floating Point Representation**: 0 10001011 01000000110100000000000.

**Q3**

**If main memory is of 1 gigabyte (1 GB) and every word is of 2 bytes, how many bits do we need to address any single word in memory?**

* **Step 1**: Calculate the number of words in memory:
  + 1 GB = 2^30 bytes.
  + Since each word is 2 bytes, the total number of words is 2^30 / 2 = 2^29 words.
* **Step 2**: Determine the number of bits needed to address 2^29 words:
  + Number of bits needed = 29 bits.

**Q4**

**How much RAM memory can a 64-bit CPU use?**

* **Step 1**: A 64-bit CPU can address 2^64 bytes of memory.
* **Step 2**: Calculate the amount of memory:
  + 2^64 bytes = 16 exabytes (EB).
* **Final Answer**: A 64-bit CPU can use 16 exabytes (EB) of RAM memory.

**Q5**

**Consider a 6-stage pipelined CPU where every stage is 40 nanoseconds. How much time does it take to execute 1000 CPU instructions if no stall cycles occur?**

* **Step 1**: Calculate the time to execute the first instruction:
  + Since it's a pipelined CPU, the first instruction takes 6 stages × 40 ns = 240 ns.
* **Step 2**: Calculate the time for subsequent instructions:
  + Each additional instruction takes 40 ns (one stage).
  + So, for 999 additional instructions: 999×40 ns=39960.
* **Final Time**:
  + Total time = 240 ns (for the first instruction) + 39960 ns (for the remaining 999 instructions) = 40,200 ns.

**Q6**

**Given a CPU frequency of 1 GHz and it executed 2 million instructions in 6 million CPU clock cycles:**

**A. What is the average value of the CPI for this CPU?**

* **Step 1**: Calculate the CPI (Cycles Per Instruction):
  + CPI = Total clock cycles / Total instructions.
  + CPI = 6,000,000 cycles / 2,000,000 instructions = 3 cycles per instruction.

**B. How long did it take to complete all the instructions?**

* **Step 1**: Convert CPU frequency to time per cycle:
  + CPU frequency = 1 GHz = 1 billion cycles/second.
  + Time per cycle = 1 ns (nanosecond).
* **Step 2**: Calculate the total time taken:
  + Total time = Total clock cycles × Time per cycle.
  + Total time = 6,000,000 cycles × 1 ns = 6,000,000 ns = 6 milliseconds.

Q7

**A CPU has an average CPI of 2. It took 1.5 seconds to execute 10 million instructions. What is the speed of this CPU?**

* **Step 1**: Calculate the total clock cycles used:
  + Total clock cycles = CPI × Total instructions.
  + Total clock cycles = 2 × 10,000,000 = 20,000,000 cycles.
* **Step 2**: Determine the CPU frequency:
  + Frequency (Speed) = Total clock cycles / Total time.
  + Frequency = 20,000,000 cycles / 1.5 seconds = 13.33 million cycles/second = 13.33 MHz.
* **Final Speed**: The speed of this CPU is approximately 13.33 MHz.

**Q8**

section .data

A db 3, 2, 3, 1, 0, 5, 7, 8, 9, 2 ; array A[10]

section .bss

i resb 1 ; variable i

section .text

global \_start

\_start:

mov ecx, 10 ; Set loop counter to 10 (for i < 10)

xor esi, esi ; Set index i to 0

loop\_start:

; Calculate 3\*i

mov eax, esi

lea eax, [eax + eax\*2] ; eax = 3\*i

; Calculate (2\*i)

mov ebx, esi

lea ebx, [ebx + ebx] ; ebx = 2\*i

; Calculate ((2\*i + 2) / 5)

add ebx, 2 ; ebx = 2\*i + 2

xor edx, edx ; Clear edx before division

mov ecx, 5 ; Divisor is 5

div ecx ; eax = quotient, edx = remainder

; Add to the previous result in eax

add eax, ebx ; eax = 3\*i + 2\*i + (2\*i+2)/5

; Add eax to A[i]

add [A + esi], al

; Increment i

inc esi

; Check if i < 10

cmp esi, 10

jl loop\_start

; Exit program

mov eax, 60 ; syscall: exit

xor edi, edi ; status: 0

syscall