Morgan Hodge

1. **A. Draw the circuit diagram of the following expression F=ABC + AB’ (A’ C’)’ [3 marks].**

Diagram

Description automatically generated

**B. Set up the truth table [8 marks]**

A picture containing table

Description automatically generated

**2. A, What decimal value does the 8-bit binary number 11011111 have if**

i) it is interpreted as an unsigned number? [1 mark]

223

ii) it is on a computer using signed-magnitude representation? [2 mark]

-95

iii) it is on a computer using one’s complement representation? [3 marks]

-32 (CHECK WITH THEO IF THIS IS NEGARTUIVE)

iv) it is on a computer using two’s complement representation? [3 marks]

33

**B. Convert the positive number Ν=1010000001011 in single precision floating point format [3 marks]**

01000101101000000101100000000000

**3. If main memory is of 64 Mbytes and every word is of 2 bytes how many bits do we need to address any single word in memory? [6 marks]**

225

**4. How much RAM memory can a 16-bit, 32bit and 64-bit CPU can use? Provide your answer in bytes. [6 marks]**

16bit = 65,536 bytes

32bit = 4,294,967,296 bytes

64bit = 18,446,744,073,709,551,616 bytes

**5. Consider a 6-stage pipelined CPU where every stage is 40nsecs. How much time does it take to execute 200 CPU instructions if no stall cycles occur? Provide the answer in nsecs. [5 marks]**

**6. Consider that the CPU clock rate is 2.2 MHz and the Program takes 1.2 million cycles to execute. What’s the CPU time (provide the answer in seconds)? [5 marks]**

0.00264 seconds

**7. Consider that a CPU supports 130 different instructions. How many bits are needed for the instruction's opcode? [5 marks]**

28 = 256, 256> 130.8 bits required

**8.**

**; AS1**

**.386 ; Specify instruction set**

**.model flat, stdcall ; Flat memory model, std. calling convention**

**.stack 4096 ; Reserve stack space**

**ExitProcess PROTO, dwExitCode: DWORD ; Exit process prototype**

**.data ; data segment**

**arrayA DWORD 3,2,3,1,7,5,7,8,9,2**

**.code ; code segment**

**main PROC ; main procedure**

**mov ebx, 0 ; this makes ebx 0 wich is the start of the c loop, ebx is now i**

**loop1:**

**mov ecx, 2 ;ecx = 2**

**mov esi, 3 ;esi = 3**

**mov eax, ebx ;if i times ebx by 2 it will no longer be a counter so i have made a copy of it**

**MUL ecx ;times i by 2**

**add eax, 1 ; add 1 to i**

**div esi**

**add eax, ebx**

**MUL esi ; times i by 3**

**mov esi, OFFSET arrayA**

**mov ecx, [ esi + TYPE arrayA \* ebx ] ;move A[i] into ecx**

**add ecx, eax ;A[i] += VAL**

**mov [ esi + TYPE arrayA \* ebx ], ecx ;move back into array**

**inc ebx ; add i**

**cmp ebx, 10 ; compare i to 10 ;**CMP PUTS INFO INTO REGISTERS THAT I DONT WORRY ABOUT

**jne loop1 ;jump if i lower to 10 ;** JNE LOOKS AT THE REGISTER AFTER, this is the end of the loop

**INVOKE ExitProcess, 0 ; call exit function**

**main ENDP ; exit main procedure**

**END main ; stop assembling**