**2’s Compliment integers represent only negative integers** (F**). The set of AND, OR, NOT gate is logically complete**(T**). In a Von Neumann machine data and instructions both reside in memory**(T). **The program counter contains the address of the next instruction to be executed**(T). **The addressing mode is a mechanism for specifying where the operand is located** (T). **The addressing mode of LDI is PC-Relative**(F). **Give the decimal value for this 2’s complement bit pattern: 1101** (*Skip all the zeros until you hit a 1. Then leave the first 1 and flip everything else*). **Ans: 00111 – (2^2+2^1+2^0)=-7.** **Give the decimal number 128 as a number in base 8.** *128/8 = 16 + 0 ;; 16/8 = 2 + 0 ;; 2/8 = 0 + 1 == (200)base8…* **What is the least negative decimal number that can be stored in an LC-3 register?** *-2^15*. **How many memory locations are in LC-3?** *2^16*. **What is the opcode for STI?** *1011*. **What is the width (in bits) of MAR and MDR in LC-3?** *MAR: 16bits MDR: 16bits).* **How one can use a single LC-3 instruction to move the value of R2 into R3?** *ADD R3, R2 #0*. **The LC-3 has no subtract instruction. Write an LC-3 Code that could perform the following operation:** **R1 = R2-R3:** *NOT R3, R3 || ADD R3, R3, #1 || ADD R1, R2, R3*. **Using only one LC-3 instruction and without changing the contents of any register, how could one set the condition codes based on the value that resides in R1?** *ADD R1, R1, #0.* **Write LC-3 instruction that clears the contents of R2?** *AND R2, R2, #0****.* The LC-3 has no OR instruction. Write an LC-3 code that could perform the following operation: *R3=R1 or R2?*** *NOT R1, R1 || NOT R2, R2 || AND R3, R1, R2 || NOT R3, R3* Immediate addressing mode implies that all operands are in registers **FALSE** In PC relative addressing the offset cannot be negative **FALSE** In the von Neumann machine architecture data and instructions can be found mixed together in memory **TRUE** The registers of a machine are part of its ISA **TRUE** Addressing modes are part of the ISA **TRUE** The .END pseudo-op is translated into machine instruction **FALSE** Assembly languages are more user friendly than machine languages -**TRUE** Normally the values of symbol table entries generated by .EXTERNAL are resolved during link-time **TRUE** The system control block or the trap vector table is a set of memory locations which contain the starting addresses of the service routines **TRUE** Invoking a subroutine automatically saves R7 which holds the return address **TRUE** At any point there are as many frames in the stack as there are pending subroutines(including main()) **TRUE** In recursion, the run-time stack keeps track of each invocation(call) of the subroutine via an activation frame. **TRUE** The LC-3 uses special I/O instructions to perform I/O.**TRUE** A computer has a 16 bit address space A[15:0]. If all addresses having bits A[15:14]=11 are reserved for I/O device registers, then the maximum number of actual word addressable memory locations is 2^16-2^14 **TRUE** Polling the I/O device is more efficient than interrupt driven I/O.**TRUE** A NOOP is an assembly language instruction that is not translate d to a machine language instruction. **TRUE** The location counter is initialized using the .ORIG assembler directive. **TRUE** The location counter for an assembly program is used during the execution of the program to keep track of the location of each instruction. **FALSE** The .FILL initializes the space it allocates **TRUE** Once the monitor finishes processing the character on the screen, DSR[15] becomes \_**1**\_. The stack pointer always points to the top of the stack. **TRUE** Every stack frame has its own frame pointer **FALSE** The .BLKW initializes the space it allocates **FALSE** Once a program reads a character from the KBDR, automatically KBSR[15] is \_\_**0**\_\_. **Addressing Modes**: **LEA**=immediate **LDR**=Base+offset **LD**=Pc-relative **LDI**=indirect **ST**=Pc-relative **STR**=base+offset **STI**=indirect. **Give an instruction in LC-3 that corresponds to the pseudo-instruction CLEAR R1, which sets R1 to 0**. AND R1, #0. **What is the LC-3 assembly language instruction for this machine code: 0110111111111111?** [opcode:01101] [DR:111] [BASEr1111] [Offset6:1111] -> LDR R7, R7, #-1**. LEA modifies the condition code bits**(T). **.END is an assembly language instruction** (F, assembly directive). **HALT is actually a TRAP instruction** (T x25) **Using operate type instructions only place the value 45 in R1.**  *AND R1, R1 #0 || ADD R1, R1, #15 || ADD R1, R1, #15 || ADD R1, R. 1, #15.* **In a Von Neumann machine data and instructions both reside in memory.** (T) **What is the opcode for GETC in LC-3**. 1111 **In LC-3 all memory can be accessed with 16 bits**. (T). **Give the decimal value for this 2’s complement bit pattern: 111111110001**. *000000001111 – 2^3 + 2^2 + 2^1 + 2^0 =15.* **Give the decimal number 119 as a number in base 5.** *5/119 = 23 + 4 || 23/5 = 4 + 3 || 4/5 = 0 + 4 (Read the remainder from bottom up = (434)base 5.* **What is the range of values that can be represented with 16 bits in two’s complement format?** *Nbits => max => -1 || Nbits => min => ()*.