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 translated 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

If x>=0 then y=y-1, else y= y+1

.ORIG x3000

LD R0, X

LD R1, Y

ADD R0,R0,#0

BRn ELSE

ADD R1,R1,#-1

BR STOP

ELSE ADD R1,R1,#1

STOP HALT

X .FILL #0

Y .FILL #2

.END