I. Process
   A) No. A program is a static set of directions, whereas the latter is a dynamic activity
      whose properties change as time progresses. A process encompasses the current
      status of the activity, called the process state.
   B) Context switch: the stopping of one process and the starting of another. The OS
      must save the registers (PC, SP, general-purpose regs) and any other state
      information in the process’ PCB.
   C) Ability so it appears that more than one program at a time is executed
   D) Program code, static data, dynamic data (stack and heap), registers (general-
      purpose registers, PC, SP, …), OS resources….

II. Machine cycle
To execute an instruction the processor must 1) Fetch the instruction from memory
2) Decode the instruction 3) execute the instruction 4) store the result back in the memory
(5) Goto step 1) These four steps refer to Machine cycle

III. von Neumann
   a) CPU, memory, stored program

   [CPU]   [Memory]   [I/O]
   bus

   b) Yes, it has a single CPU and a general-purpose memory that holds both program
      and data

IV. Two free points up for grabs …

V. Number conversion table
   0001 0001
   11
   77
   4D
   165
   1010 0101
VI. Ada

Part a. The procedure test will not compile. Line 12 in the test procedure raises an error because the expected value is an integer, but a float is being passed.

The modification required in Line 12 in order to keep the same order of parameters is

```ada
    Ada_Read_1(Input_Float=>Test_Float,
               Input_Integer=>Test_Integer);
```

Part b. The value of count displayed is 25. The inner loop will only execute if I is odd. Hence the count is incremented $5 \times 5 = 25$ times.

Part c.

```ada
with Ada.Text_Io;  -- Program to compute the first n fibonacci numbers
with Ada.Integer_Text_Io;
use Ada.Text_Io;   -- Programmer : Jayakanth Srinivasan
use Ada.Integer_Text_Io;   -- Date Modified : Mar 08, 2003
-- The program does not take into consideration integer overflow
procedure Fibonacci is
    N                : natural:= 0;  -- user input for number of fibo numbers
    First_Number     : Integer  := 0;
    Second_Number    : Integer  := 1;
    Temporary_Number : Integer;  -- used to store the generated numbers before display
    begin
    Put("Please enter the number of fibonacci numbers > 0");
    Get (N);
    Skip_Line;
    New_Line;
    case N is
    when 0 => -- handles the case when user enters 0
        Put("No numbers to display");
        New_Line;
    when 1 =>-- when the user wants to see the first fibo number
        Put ("The fibonacci number is :");
        New_Line;
    when 2=-- when the user asks for the first two fibo numbers
        Put ("The first fibonacci number is ");
        New_Line;
        Put(Natural'Image(First_Number));
        New_Line;
    ```
33.          Put("The fibonacci numbers are :" );
34.          Put(Natural'Image(First_Number));
35.          Put(",");
36.          Put(Natural'Image(Second_Number));
37.          New_Line;
38.       when others => -- when the user asks for more than 2 fibo numbers
39.          Put("The fibonacci numbers are :" );
40.          Put(Natural'Image(First_Number));
41.          Put(",");
42.          Put(Natural'Image(Second_Number));
43.          for I in 3 .. N loop
44.             -- compute the next fibo number and display to user
45.             Temporary_Number := First_Number + Second_Number;
46.             Put(",");
47.             Put(Natural'Image(Temporary_Number));
48.             -- update the first and second numbers
49.             First_Number := Second_Number;
50.             Second_Number := Temporary_Number;
51.          end loop;
52.          New_Line;
53.          end case;
54.          end Fibonacci;

VII. Machine language
a. Algorithm

Integer division is defined as follows: when m is divided by n, we obtain two integers, q and r, called quotient and remainder, respectively, with r between 0 and n-1, such that the equation m = n*q + r is satisfied.

1. Assume that the numbers are m, n and the operation is m/n
2. quotient :=0, remainder :=0;
3. if (n = 0) Goto step 6
4. if m < n then
   a. Set remainder := m;
   b. Goto step 6
5. loop while m >= n
   a. increment quotient
   b. m:= m-n
   c. if m < n then set remainder := m;
6. store the values of quotient and remainder
7. halt

; Program to perform integer division on two positive numbers.
; Programmer : Jayakanth Srinivasan
; Date: Mar 08, 2003
; assumes that numbers are in two's complement integers
; values limited to 0 - 127

; code segment
    load R0, 0;
    load R1, 0; set the quotient to zero
    load R2, 0; set the remainder to zero
    load R3, [divisor]; load the divisor
    load R4, [dividend]; load the dividend
    load R5, 1; increment for quotient and computing 2's complement
    jmpEQ R3=R0, store_values; if divisor is zero, set quotient and
        ; remainder to zero
    load R6, 0xff; mask for flipping the bits
    xor R7, R3, R6; flip the 0's and 1's in the divisor
    addi R7, R7, R5; add 1 to the flipped bits to get the
        ; 2's complement of the divisor
    move R0, R3;
    jmpLE R4<=R0, set_remainder; if dividend less than divisor

division_loop:
    move R0, R4;
    jmpEQ R3=R0, final_add; if its a
    addi R4, R4, R7; subtract the divisor from the dividend
    addi R1, R1, R5; increment the quotient counter
    move R0, R4; copy dividend value into R0
    jmpLE R3<=R0, division_loop; continue if dividend> divisor

set_remainder:
    move R2, R4; copy remainder into R2
    jmp store_values;

final_add:
    addi R1, R1, R5;

store_values:
    store R1, [quotient]; store the quotient
    store R2, [remainder]; store the remainder

; data segment

dividend: db 6; m in m/n
divisor : db 96; n in m/n
quotient: db 0;
remainder: db 0;

VIII. Multiple-choice questions

B, D, C, C, B, B, C, A, D