**Zoratti Marco, 5562866**

C1) Il numero 11000000000000000000000000000000 è un numero binario formato da 32 cifre (o bit).

C2) Dato il numero 11000000000000000000000000000000:

(2^0 \* 0 = 0) + (2^1 \* 0 = 0) + (2^2 \* 0 = 0) + ... + (2^30 \* 1 = 1073741824) + (2^31 \* 1 = 2147483648) = 1073741824 + 2147483648 = 3221225472

A1) Il programma allegato nel file di consegna chiede l'inserimento di un numero intero e stampa "PIPPO" se esso è pari o "PLUTO" se è dispari.

A2) Un possibile programma equivalente in c++ (utilizzando solo il main):

#include <iostream>

int main() {

int a;

std::cout << "Inserisci un numero intero: ";

std::cin >> a;

if(a%2 == 0)

std::cout << "\nPIPPO";

else

std::cout << "\nPLUTO";

return 0;

}

A3)

.data

msgi: .asciz "Scegli un valore "

msgp: .asciz "Il numero piu grande è: "

msgd: .asciz "Il numero piu piccolo è: "

V: #Alloco un vettore di 10 posizioni

.word 0,0,0,0,0,0,0,0,0,0

.text

la t1, V #carica in t1 l'array (l'elemento con index = 0)

la t2, V #carica nuovamente l'array in t2

addi t2, t2, 40 #aggiungi 40 byte all'indirizzo dell'array, cosi da avere l'inidirzzo finale

#Ciclo di memorizzazione dei dati di input nell'array

memorizzazione: li a7, 4 #Stampa una stringa

la a0, msgi

ecall

li a7, 5 #Leggi un valore

ecall

#in questo momento, a0 contiene il valore di input

sw a0, 0(t1) #memorizza l'input in v[0]

addi t1, t1, 4 #passa al prossimo elemento dell'array

bne t1, t2, memorizzazione

li t3, 1000000 #carica il minimo

li t4, -1000000 #carica il massimo

addi t1, t1, -40 #fai tornare t1 al valore iniziale

#salva il numero minimo in t3

salva\_minimo: mv t3, t1

#salva il numero massimo in t4

salva\_massimo: mv t4, t1

trova\_minimo\_massimo:

bgt t1, t4, salva\_massimo

blt t1, t3, salva\_minimo

addi t1, t1, 4 #passa al prossimo elemento dell'array

bne t1, t2, trova\_minimo\_massimo

#mostra popup per il numero piu piccolo

li a7, 56 #Stampa una stringa

la a0, msgd

mv a1, t3

ecall

#mostra popup per il numero piu grande

li a7, 56 #Stampa una stringa

la a0, msgp

mv a1, t4

ecall