Politecnico di Milano - Dipartimento di Elettronica e informazione

Prof. Mauro Negri

Fondamenti di Informatica **27 febbraio 2013**

I appello bianco

Matricola _____ Cognome_____ Nome_____

Durata prova: 2 ore

Istruzioni

LE RISPOSTE DEVONO ESSERE SCRITTE SU QUESTO PLICO (anche sul retro dei fogli)

Non separare questi fogli. Si può utilizzare la matita.

L'uso di cellulari, libri, eserciziari, appunti o calcolatrici durante lo svolgimento della prova comporta l'annullamento della prova.

Non è possibile uscire dall'aula durante la prova tranne in casi eccezionali.

Esercizio 1 (5 punti) _____

Esercizio 2 (5 punti) _____

Esercizio 3 (15 punti) _____

Esercizio 4 (5 punti) _____

Punteggio totale (30 punti) _____

Esercizio 1

1. Indicare la rappresentazione in complemento a 2, utilizzando 10 bit, dei numeri:
   A = +59 (base 10) 0000111011
   B = - 81 (base 10) 0001010001 -> 1110101111

2. Eseguire la somma A+B in binario dei numeri interi A= -5 e B= -30 codificati in complemento a 2 su 6 bit. Mostrare il risultato in binario prodotto dall'unità aritmetico-logica e indicare quale sia il numero decimale corrispondente al numero binario prodotto (usare sempre 6 bit). Infine riportare il valore dei bit di carry (riporto) e di overflow dopo l'operazione dando breve motivazione del valore assegnato.

   111011 A
   100010 B
   _____
   (1)011101 Risultato binario

   Corrispondente valore decimale (base 10): +29
   bit carry: 1
   bit overflow: 1
   Breve motivazione di carry e overflow:
   -carry 1 perché 2 negativi generano sempre riporto nella somma
   -overflow perché primo bit operandi discorde da primo bit risultato

3. Indicare il valore in base 8 e 16 corrispondente al numero -75 in base 10

   base 8 = -113 base 16 = -4B

4. Indicare quale numero decimale rappresenta la seguente sequenza di bit codificata in complemento a 2:
   1111.0110.0010
   -158

5. Data la definizione della variabile double v= - 55,6, descrivere la sua rappresentazione in virgola mobile in base allo standard IEEE754 semplice precisione FP64 descritto a lezione;
   a) Definire prima il numero nel formato normalizzato intermedio (TRONCARE parte decimale alla terza cifra decimale):
   $-1{,}737 * 2^5$

   b) Codificare poi il numero normalizzato usando il formato FP64:
   S (1 bit di segno): 1
   M (52 bit per la mantissa – fermarsi ai primi 6 bit): 101111
   E (11 bit per l'esponente): 10000000100

   c)Indicare quale sia il vero valore (in base 10) descritto dalla sequenza di bit memorizzata in S, M ed E del punto b) precedente (si consiglia di esprimere la mantissa in termini frazionari).
   -55,5

6. Si consideri l'intervallo di numeri reali compresi tra 1.050.000 e 2.000.000 e la loro codifica in FP64. Scrivere l'espressione che determina la distanza minima tra due numeri reali (vicini) realmente rappresentati dalla codifica specificata.

I numeri in questo intervallo sono del tipo $1{,}xx * 2^{20}$ quindi la variazione del bit meno significativo (52esimo) comporta una distanza in 1,xx di 2^{-52} , pertanto la distanza tra i due numeri reali adiacenti sarà:

$$2^{-52} * 2^{20}$$

Esercizio 2.

2.a Tradurre il seguente programma C in assembler

```
#include <stdio.h>
int ris=1, z, v[2];

int f(int x, int *y)
    { struct el {int c1; int c2;};
      struct el v= {3, 5};
      *y=v.c2;
       y++;       //istruzione 1
      *y=v.c1;
      return v.c1;
    }
int main ()
    { …          z=f(ris, v));   }
```

Utilizzare le seguenti istruzioni assembly

[W:] ADD oper1,oper2	somma in oper2
[W:] MOV oper1, oper2	copia in oper2
[W:] RTS	ritorno da funzione
[W:] JTS oper	invocazione sottoprogramma
[W:] EXIT	termina esecuzione programma

Modalità di indirizzamento operandi delle istruzioni:

- Ri o SP — contenuto del registro Ri o SP
- (Ri) o (SP) — contenuto della parola di memoria il cui indirizzo è nel registro Ri o SP
- #X — valore del simbolo X
- X — contenuto della parola di memoria con etichetta X

ATTENZIONE:

- I registri R1,…, R6 (non R0 e SP) possono essere usati senza salvare precedentemente il loro contenuto.
- Deve essere allocato lo stack (1000 parole) e inizializzato correttamente il registro SP.

```
RIS: .RES 1
Z: .RES 1
V: .RES 2
STACK: .RES  1000
MAIN: MOV #STACK, SP
      ADD #999, SP
      MOV #1, RIS

      ADD #-1, SP   //risultato

      MOV RIS, (SP)      //parametri
      ADD #-1, SP
      MOV #V, (SP)
      ADD #-1, SP

      JTS F

      ADD #3, SP
      MOV (SP), Z
      EXIT

F:    MOV R0, (SP)
      ADD #-1, SP
      MOV SP, R0
      ADD#-2,SP

      MOV #3, (R0)
      MOV R0, R1
      ADD #-1, R1
```

```
MOV #5, (R1)   //V.C2

MOV R0, R2
ADD #3, R2   //R2 punta a cella con indirizzo di v globale
MOV (R2), R3       //R3 contiene indirizzo di v globale
MOV (R1), (R3)     //*y=v.c2

ADD #1, (R2)//y++

MOV (R2), R3       //R3 contiene indirizzo di v globale
MOV (R0), (R2)   //*y=v.c1

MOV R0,R4
ADD #5,R4
MOV (R0), (R4)     // copia v.c1 di return nella prima parola dello stack

ADD # 3, SP
MOV (SP), R0
RTS
END MAIN
```

Esercizio 2.b Visualizzare con una figura lo stato dello stack all'atto dell'esecuzione dell'istruzione 1, riportando la parola puntata da R0 e SP e il contenuto logico di ogni parola allocata per il RDA della funzione f.

	<- SP
V.C2	
V.C1	<- R0
R0	
IND.RITORNO	
INDIRIZZO DI V Y	
RIS X	
Valore di ritorno	

Esercizio 3. Siano date le seguenti definizioni
struct el {int numero; int posizione;};
struct din {struct el dati; struct din * prox);
struct din *LIS=NULL;

Scrivere un **programma** in C che definisce le seguenti funzioni:

a) La funzione **Ricerca** che riceve come parametri un numero intero N ≥0, una posizione (-1 o un valore ≥ 0) e il descrittore di un file aperto (di tipo FILE *) che contiene una serie di numeri interi (si consiglia un file binario). La funzione verifica la correttezza dei parametri e in caso contrario si ferma e ritorna -1. La funzione cerca nel file il numero N: nel caso in cui pos=-1 la funzione si ferma al primo che incontra e ritorna, come parametro, l'indirizzo del primo byte che memorizza il numero trovato nel file. Se la posizione è ≥ 0 ricerca se esiste nel file il numero N memorizzato all'indirizzo ricevuto nel parametro posizione e una volta trovato ricerca il prossimo valore N memorizzato nel file del quale ritorna l'indirizzo del primo byte. Se la ricerca non ha successo per qualsiasi motivo restituisce -1.

b) La funzione main chiede al terminale di inserire il valore di N finché non viene introdotto un numero ≥ 0. Poi, utilizzando le precedenti funzioni ricerca nel file FILE.DAT (se non esiste termina l'esecuzione) tutte le istanze del numero N presenti nel file, estraendone per ognuno la posizione nel file e inserendo ogni coppia <numero, posizione> nella lista LIS. Se l'inserimento in lista fallisce o quando tutte le coppie sono state caricate in lista il programma termina la propria esecuzione.

Si supponga che i valori letti al terminale siano numeri interi che non generano overflow.

```
#include <stdio.h>
#include <stdlib.h>
struct el {int numero; long int posizione;};
struct din {struct el dati; struct din *prox;};
struct din *LIS=NULL;

int Ricerca (int N, FILE *id, int pos)
{int val; long int newpos;int trovato;
 if (N <0) return -1;
 if (pos <-1) return -1;
 IF (ID==NULL) RETURN -1;
    if (pos==-1)
       {rewind(id); trovato=0;
         fread(&val, sizeof(int), 1 , id);
         while ((!trovato) && (!feof(id)))
           { if (val==N) trovato=1;
                  newpos=ftell(id)- sizeof(int); //cercaprimoN
             fread(&val, sizeof(int), 1 , id);
            }
         if (trovato) return newpos; else return -1;//anche per file vuoto
        }
    else { //non controllato se pos non esite tramite controllo valore
           //ritornato dalla fseek
             fseek(id, pos, SEEK_SET); //cerca N in POS
             fread(&val, sizeof(int), 1 , id);
             if (val==N)
                  {trovato=0; fread(&val, sizeof(int), 1 , id);
                   while ((!trovato) && (!feof(id)))
                    { if (val==N) {trovato=1; newpos=ftell(id)- sizeof(int); }
                      else fread(&val, sizeof(int), 1 , id);
                    }
                  if (trovato) return newpos; else return -1;//cerca prossimo N
                  }
             else return -1;
           }
}


int main()
{  FILE *filid; int N, pos; struct el temp;
    do {  printf("introdurre N"); scanf("%d",&N); }while (N<=0);
    filid = fopen("DATI.DAT", "rb");
    if (filid==NULL) {  printf("\nErrore nell'apertura del file! "); exit(0); }
     pos= Ricerca (N, filid, -1);
     while (pos!=-1)
    { temp.numero= N; temp.posizione=pos ;
                if (Carica(temp, &LIS)==0) {printf("finito"); exit(0); }
          pos = Ricerca (N, filid, pos);
    }
     printf("finito");       fclose(filid);
}
```

Esercizio 4. Si supponga di eseguire il seguente programma

```
#include <stdio.h>
#include <stdlib.h>
int a,b=116, status, log1; pid_t pid;
int f(int *e) {*e=120;return 112;}
int g (int *e) {a=177; *e=1110;}

int main ()
{pid=fork();
 //posizione T1
 printf("\nT1: pid=%i a=%d, b=%d, status=%d,log1=%d", pid,a,b,status,log1);fflush(stdout);
 if (pid == 0)
     {log1= f(&b);
      //posizione T2
      printf("\nt2: pid=%d, a=%d, b=%d, status=%d,log1=%d", pid,a,b,status,log1);fflush(stdout);
      pid=fork();
      if(pid!=0)
         { pid=wait(&status);
          //posizione 3
          printf("\nT3: pid=%d, a=%d, b=%d, status=%d,log1=%d", pid,a,b,status,log1);fflush(stdout);
         exit(1);
         }
      else exit(10);
     }
 else
  { g(&b);
   //posizione T4
   printf("\nT4: pid=%d, a=%d, b=%d, status=%d,log1=%d",pid, a,b,status,log1);fflush(stdout);
  a=40;
  }
}
```

e di sapere che il processo padre ha pid=1000, il primo figlio pid=1001 e il secondo figlio pid=1002 e che la le stampe a terminale avvengono con la seguente sequenza:

- Stampa posizione T1 (padre)
- Stampa posizioneT4
- Stampa posizione T1 (primo figlio)
- Stampa posizione T2
- Stampa posizione T3

A questo punto riportare nelle seguenti tabelle il valore che le variabili hanno nei 3 processi quando un'esecuzione passa dalle quattro posizione T1, T2,T3,T4. Usare la seguente convenzione per riempire la tabella

NE: quando il processo non esiste in una delle quattro posizioni;

X: se la variabile esiste ma non è stata ancora inizializzata;

?: se non si è in grado di valutare con certezza il valore nella posizione richiesta

La soluzione considera anche quanto non richiesto dall'esercizio, ossia il passaggio del primo figlio da T1, a scopo di esercizio.

Processo Padre

	Pid	A	B	Status	log1
Posizione T1(padre)	1001	X	116	X	X
Posizione T1 (figlio)	?	?	?	?	?
Posizione T2	?	?	?	?	?
Posizione T3	?	?	?	?	?
Posizione T4	1001	177	1110	x	X

Il padre passa per primo dalla posizione T1 subito dopo la fork che restituisce nel pid l'id del processo figlio. Per semplicità si passa a considerare la posizione T4 dove il padre proseguendo la propia esecuzione avrà invocato la funzione g che cambia alcune sue variabili. Consideriamo la posizione T1 quando vi passa il primo processo figlio. Dalla sequenza di stampa si evince che il padre è già passato da T4, ma poiché il padre non ha wait potrebbe essere ancora in T4, aver eseguito a=40 o aver già terminato la propria esecuzione (la più probabile, ma non sicura), pertanto i valori sono indeterminabili. A maggior ragione ciò accade quando i figli passano dalle altre posizioni.

I processo figlio

	Pid	A	B	Status	log1
Posizione T1(padre)	0	X	116	X	X
Posizione T1(figlio)	0	X	116	X	X
Posizione T2	0	X	120	X	112
Posizione T3	1002	X	120	2560	112
Posizione T4	0	X	116	X	X

Il primo figlio esiste quando il padre esegue la prima istruzione dopo la fork (Posizione T1 padre), ma dalla sequenza di stampa si capisce che il figlio non ha ancora fatto nulla, infatti prima che il figlio passi da T1 il padre è già in/oltre T4, pertanto il figlio avrà il valore delle variabili del padre in T1 ad eccezione del pid che sarà 0. La stessa condizione si avrà quando il padre passa da T4 perché il figlio è ancora fermo e quando il figlio passa finalmente da T1. Il figlio proseguendo nell'esecuzione attiva la funzione f e passa da T2 dove cambiano solo due variabili. Infine il figlio crea un proprio figlio (ciò altera la propria variabile pid) e passa da T3 subito dopo la wait che altera lo status (avrà ricevuto il 10 del secondo figlio moltiplicato 256).

II processo Figlio

	Pid	A	B	Status	log1
Posizione T1(padre)	NE	NE	NE	NE	NE
Posizione T1 (figlio)	NE	NE	NE	NE	NE
Posizione T2	NE	NE	NE	NE	NE
Posizione T3	NE	NE	NE	NE	NE
Posizione T4	NE	NE	NE	NE	NE

Il secondo processo figlio quando il padre passa da T1 e T4 non esiste e la stessa situazione si verifica quando anche il primo figlio passa da T1 e da T2 che precede la seconda fork. Quando il primo processo figlio passa da T3 significa che il primo figlio ha ricevuto notifica della exit del II figlio che quindi è stato distrutto, pertanto il secondo figlio non esiste più in T3