[T05] Esercitazione del 25 marzo 2022

Istruzioni per l'esercitazione:

- Aprite il form di consegna in un browser e loggatevi con le vostre credenziali `uniroma1`.
- Scaricate e decomprimete sulla scrivania il codice dell'esercitazione. Vi sarà una sotto-directory separata per ciascun esercizio di programmazione. Non modificate in alcun modo i programmi di test `*_main.c`.
- Rinominare la directory chiamandola `cognome.nome`. Sulle postazioni del laboratorio sarà `/home/biar/Desktop/cognome.nome/`.
- È possibile consultare appunti/libri e il materiale didattico online.
- Rispondete alle domande online sul modulo di consegna.
- **Finiti gli esercizi**, e non più tardi della fine della lezione:
  - **zippate la directory di lavoro** in `cognome.nome.zip` (`zip -r cognome.nome.zip cognome.nome/`).
- **Per consegnare**:
  - inserite nel form di consegna come autovalutazione il punteggio di ciascuno dei test forniti (inserite zero se l'esercizio non è stato svolto, non compila, o dà errore di esecuzione).
  - fate **upload** del file `cognome.nome.zip`.
- **Se avete domande** accedete a Zoom agli orari stabiliti per l'esercitazione, accedendo con la **mail istituzionale** uniroma1.it dello studente. Troverete online i docenti ed il tutor del corso. In alternativa, scrivete via mail ai docenti.
- **Suggerimento** Non traducete direttamente da C a IA32, ma scrivere prima una versione C intermedia `E1/e1_eq.c` equivalente a quella di partenza, ma più semplice da tradurre in assembly. Testatela con il main di prova prima di passare a scrivere la versione `.s`. E' inutile tradurre la versione C equivalente se è errata!

Per maggiori informazioni fate riferimento al regolamento delle esercitazioni.

Esercizio 1 (Inserimento in testa in una lista collegata)

Tradurre nel file `E1/e1.s` la seguente funzione C contenuta in `E1/e1.c` che aggiunge un elemento in testa una lista collegata:

```c
#include <stdlib.h>
#include "e1.h"

int list_add_first(node_t **l, short elem) {
    node_t *p = *l;
    node_t *n = malloc(sizeof(node_t));
    if (n == NULL) return -1;           // allocation error
    n->elem = elem;
    n->next = p;
    *l = n;
    return 0;
}
```

Usare il main di prova nella directory di lavoro E1 compilando con `gcc -m32 e1_main.c e1.s -o e1`.

Esercizio 2 (Uguaglianza di liste)

Tradurre nel file E2/e2.s la seguente funzione C contenuta in E2/e2.c che, date due liste collegate, restituisce 1 se sono uguali e 0 altrimenti. Riferirsi alla dichiarazione della struttura node_t dell'Esercizio 1.

```
int list_equal(const node_t *l1, const node_t *l2) {
    while (l1!=NULL && l2!=NULL) {
        if (l1->elem != l2->elem) return 0;
        l1 = l1->next;
        l2 = l2->next;
    }
    return l1==NULL && l2==NULL;
}
```

Suggerimento: usare le istruzioni SETcc.

Usare il main di prova nella directory di lavoro E2 compilando con `gcc -m32 e2_main.c e2.s -o e2`.

Esercizio 3 (strpbrk alla vaccinara)

Si richiede di scrivere un'implementazione della funzione standard C strpbrk:

```
char *my_strpbrk(const char *s1, const char *s2);
```

La funzione deve restituire il puntatore alla prima occorrenza in `s1` di un qualsiasi carattere presente nella stringa `s2` oppure `NULL` se alcun carattere di `s2` appare in `s1` prima che `s1` stessa termini.

Esempio: Se `s1 = "Once again, this is a test"` e `s2="ftir"`, `my_strpbrk` deve restituire il puntatore alla prima occorrenza in `s1` di un qualsiasi carattere nella stringa `s2`, cioè il puntatore alla `i` di "again". La chiamata `my_strpbrk("Once again, this is a test",":#F")` deve invece restituire `NULL`. Ovviamente non è accettabile chiamare direttamente la `strpbrk` predefinita della `libc` (non imparereste nulla).

Esercizio 4 (Domande)

1. Assumendo che il registro %ecx contenga il valore 0xF0F0F0F0, dopo aver eseguito l'istruzione "shll $4, %ecx" quale delle seguenti affermazioni risulta vera:
   - A. Il bit più significativo di %ecx è 1
   - B. Il bit meno significativo di %ecx è 1
   - C. %cl contiene il valore 0
   - D. %ch contiene il valore 0
   - E. Se si esegue l'istruzione "sarl $4, %ecx" si riporta il valore di %ecx a 0xF0F0F0F0
2. Assumendo che il registro %ecx contenga il valore 0x00FF00FF, quale delle seguenti affermazioni risulta vera:
   - A. Se eseguiamo "sarl $3, %ecx" otteniamo in %ecx un numero più grande rispetto a prima
   - B. Se eseguiamo "sarl $8, %ecx" oppure "shrl $8, %ecx" otteniamo lo stesso valore in %ecx
   - C. Se eseguiamo "sarl $31, %ecx" otteniamo in %ecx un numero diverso da 0
   - D. Se eseguiamo "sarb $1, %cl" il valore contenuto in %ecx cambia
   - E. Nessuna delle precedenti
3. Quale delle seguenti istruzioni *NON* calcola lo stesso risultato di "shll $1, %eax":
   - A. imul $2, %eax

- B. addl %eax, %eax
- C. leal (%eax, %eax), %eax
- D. leal 2(%eax), %eax
- E. sall $1, %eax

4. Siano %eax=33 (decimale), %edx=0 (decimale) e %ecx=10 (decimale), quale delle seguenti affermazioni risulta vera:
   - A. Per calcolare %eax diviso 10 posso eseguire "idivl %eax" oppure "sarl $10, %eax"
   - B. "idivl $3" è un'istruzione valida e calcola "%edx:%eax" diviso 3
   - C. Non posso usare "sarl" per calcolare %eax diviso 2, perché %eax contiene un valore che non è una potenza di 2
   - D. "idivl %ecx" scrive il valore 3 nel registro %eax e nel registro %edx
   - E. Nessuna delle precedenti
5. Data la struct C "struct s { char x, short * y; int z;}", quanti byte (sizeof) sono necessari per memorizzarla in stack?
   - A. 7
   - B. 8
   - C. 10
   - D. 12
   - E. 14
   - F. 16
6. Data la struct C "struct s { char x, short y; char z}", quanti byte di padding sono presenti nella struct?
   - A. 0
   - B. 1
   - C. 2
   - D. 3
   - E. 4
7. Data la struct C "struct s { char x, int y; char * z; short q; char p; }", qual è l'offset del campo p?
   - A. 8
   - B. 10
   - C. 12
   - D. 14
   - E. 16
8. Considerata la costante -72, quale tra le seguenti affermazioni risulta essere vera?
   - A. La sua rappresentazione binaria (8 bit) è 0100 1000
   - B. La sua rappresentazione binaria (16 bit) è 1111 1111 1011 0110
   - C. La sua rappresentazione binaria (16 bit) è 0000 0000 1011 0110
   - D. La sua rappresentazione binaria (16 bit) è 1111 1111 0100 1000
   - E. Se gli viene sommato 1 la sua rappresentazione binaria (16 bit) è 1111 1111 1011 1000
   - F. Nessuna delle precedenti