[T02] Esercitazione del 4 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/studente/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`.
  - **importante**: fate logout dal vostro account Google!
- **Prima di uscire**:
  - eliminate dal desktop la directory creata (`rm -rf cognome.nome`).
  - firmate il **foglio presenze**.
  - rimettete a posto eventuali **sedie** prese all'ingresso dell'aula!

Esercizio 1 (Calcolo di espressioni)

Tradurre nel file `E1/e1.s` la seguente funzione C contenuta in `E1/e1.c`, **senza semplificare l'espressione manualmente**:

```
int f() {
    int x = ((2+3)*(4-2)-(2+3))*3;
    return x + 1;
}
```

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

Esercizio 2 (Calcolo di espressioni con un parametro)

Tradurre nel file `E2/e2.s` la seguente funzione C contenuta in `E2/e2.c` che calcola un polinomio a valori interi:

```
int f(int x) {
    return 2*x*x-7*x+1;
}
```

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

Esercizio 3 (Calcolo di espressioni con due parametri)

Tradurre nel file E3/e3.s la seguente funzione C contenuta in E3/e3.c che calcola un polinomio a valori interi:

```
int f(int x, int y) {
    return (x+y)*(x-y);
}
```

Usare il main di prova nella directory di lavoro E3 compilando con `gcc -m32 e3_main.c e3.s -o e3`

Esercizio 4 (Palestra C)

Scrivere nel file E4/e4.c la vostra versione personale della funzione della libreria standard libc strcat che appende la stringa `src` alla stringa nel buffer `dest` e restituisce `dest`. Il prototipo della funzione da realizzare è il seguente:

```
char *my_strcat(char *dest, const char *src);
```

Usare il main di prova nella directory di lavoro E4 compilando con `gcc e4_main.c e4.c -o e4`

Esercizio 5 (Domande)

Rispondere ai quiz riportati nel form di consegna di consegna.

Soluzioni

Esercizio 1 (Calcolo di espressioni)

Versione C equivalente, creato a partire dall'alberto sintattico dell'espressione (come visto a lezione):

`f_eq.c`

```
int f() {  // codice C equivalente
    int a = 2;
    a = a + 3;
    int c = 4;
    c = c - 2;
    a = a * c;
    c = 2;
    c = c + 3;
    a = a - c;
    a = a * 3;
    a = a + 1;
    return a;
}
```

Versione IA32:

`f.s`

```
.globl f

f:
    movl $2, %eax    #     int a = 2;
    addl $3, %eax    #     a = a + 3;
```

```
    movl $4, %ecx    #     int c = 4;
    subl $2, %ecx    #     c = c - 2;
    imull %ecx, %eax #     a = a * c;
    movl $2, %ecx    #     c = 2;
    addl $3, %ecx    #     c = c + 3;
    subl %ecx, %eax  #     a = a - c;
    imull $3, %eax   #     a = a * 3;
    incl %eax        #     a = a + 1;
    ret
```

Esercizio 2 (Calcolo di espressioni con un parametro)

Versione C equivalente, creato a partire dall'alberto sintattico dell'espressione (come visto a lezione):

```
int f(int x) {
    int a = x;
    a = a * a;
    a = a * 2;
    int c = x;
    c = c * 7;
    a = a - c;
    a = a + 1;
    return a;
}
```

Versione IA32:

```
.globl f

f:
    movl 4(%esp), %eax  # int a = x;
    imull %eax, %eax    # a = a * a;
    imull $2, %eax      # a = a * 2;
    movl 4(%esp), %ecx  # int c = x;
    imull $7, %ecx      # c = c * 7;
    subl %ecx, %eax     # a = a - c;
    incl %eax           # a = a + 1;
    ret                 # return a;
```

Esercizio 3 (Calcolo di espressioni con due parametri)

Versione C equivalente, creato a partire dall'alberto sintattico dell'espressione (come visto a lezione):

```
int f(int x, int y) { // x <-> c, y <-> d
    int c = x;
    int d = y;
    int a = c;
    a = a + d;
    c = c - d;
    a = a * c;
    return a;
}
```

Versione IA32:

```
.globl f

f:
    movl 4(%esp), %ecx  #     int c = x;
```

```
    movl 8(%esp), %edx  #     int d = y;
    movl %ecx, %eax     #     int a = c;
    addl %edx, %eax     #     a = a + d;
    subl %edx, %ecx     #     c = c - d;
    imull %ecx, %eax    #     a = a * c;
    ret                 #     return a;
```

Esercizio 4 (Palestra C)

```c
char *my_strcat(char *dest, const char *src) {
        char* d = dest;
        while (*dest) dest++;
        while (*src) *dest++ = *src++;
        *dest = 0;
    return d;
}
```

Esercizio 5 (Domande)

Domanda 1) Quale dei seguenti frammenti di codice potrebbe essere scritto in linguaggio macchina?

- `55 23 C3 D3 00 00 00 C3` [corretto, il linguaggio macchina è una sequenza di numeri]
- `movl $2, %eax`
- `while(i<y) i++;`
- nessuno dei precedenti

Domanda 2) Quanti bit servono per rappresentare il numero esadecimale 0xDEADBEEF?

- 32 [corretto, ogni cifra esadecimale corrisponde a 4 bit]
- 8
- 24`
- 16

Domanda 3) Il comando `gcc hello.s -o hello` attiva i seguenti stadi della toolchain di compilazione:

- assemblatore e linker [corretto, hello.s va prima assemblato per creare il file oggetto `.o` e poi il tutto va linkato per generare il file eseguibile `hello`]
- assemblatore
- linker
- preprocessore, compilatore e assemblatore

Domanda 4) Il comando `gcc hello.o -o hello` attiva i seguenti stadi della toolchain di compilazione:

- linker [corretto, si parte già da un modulo oggetto che è stato già assemblato, rimane solo da creare l'eseguibile `hello` con il linker]
- preprocessore
- assemblatore e linker`
- preprocessore, compilatore e assemblatore

Domanda 5) Fra i seguenti, qual è il tipo primitivo C con la sizeof minore che consente di rappresentare il numero 256?

- `short` [corretto, 256 è 2^8, quindi sevono > 8 bit per rappresentare il valore]

- `char`
- `unsigned char`
- `nessuno dei precedenti`

Domanda 6) Per quale operatore bit a bit OP si ha che `0x13 OP 0x21 == 0x32?

- `^` (XOR) [corretto, tradotto in binario si ha 0001 0011 xor 0010 0001 = 0011 0010]
- `~` (NOT)
- `&` (AND)
- `|` (OR)

Domanda 7) Dati: char s[]=”hello”; int a=sizeof(s), b=strlen(s), c=sizeof(“hello”); quale delle seguenti affermazioni è vera? Assumere puntatori a 64 bit.

- `a=6, b=5, c=6` [corretto, sia la variabile `s` che il letterale `"hello"` denotano un array di 6 byte che include il terminatore zero della stringa. Invece `strlen` restituisce il numero di caratteri escluso il terminatore]
- `a=6, b=5, c=5`
- `a=5, b=5, c=5`
- `a=6, b=5, c=8`