Sistemi di Calcolo (A.A. 2015-2016)
Corso di Laurea in Ingegneria Informatica e Automatica
Sapienza Università di Roma

C

Esame del 18/2/2016 (non esonerati) – Durata 1h 30'

Esercizio 1 (IA32)

Si traduca in assembly IA32 la seguente funzione C che concatena una stringa `b` a una stringa `a` scrivendo un modulo `es1.s`:

```
void mystrcat(char* a, const char* b) {
    while (*a) a++;
    while (*b) *a++ = *b++;
    *a = 0;
}
```

Per i test, usare il seguente programma di prova `es1-main.c`:

```
#include <stdio.h>

void mystrcat(char* a, const char* b);

int main() {
    char s1[11] = "Han\0------";
    char s2[17] = "Obi-Wan\0--------";
    char s3[19] = "Anakin Skywalker\0-";

    mystrcat(s1, " Solo");
    printf("mystrcat(s1, \" Solo\")=\"%s\" [corretto=\"Han Solo\""
           " - terminatore %s]\n",
           s1, s1[8]==0 && s1[9]=='-' ? "OK" : "ERR");

    mystrcat(s2, " Kenobi");
    printf("mystrcat(s2, \"Kenobi\")=\"%s\" [corretto=\"Obi-Wan "
           "Kenobi\" - terminatore %s]\n",
           s2, s2[14]==0 && s2[15]=='-' ? "OK" : "ERR");

    mystrcat(s3, "");
    printf("mystrcat(s3, \"\")=\"%s\" [corretto="
           "\"Anakin Skywalker\" - terminatore %s]\n",
           s3, s3[16]==0 && s3[17]=='-' ? "OK" : "ERR");

    return 0;
}
```

Generare un file eseguibile `es1` compilato con `gcc –m32`.

Esercizio 2 (IA32)

Si traduca in assembly IA32 la seguente funzione C in un modulo `es2.s`:

```
int df(int x, int y);

void update(int* v, int n) {
    while (n--) v[n] = df(50, v[n]);
}
```

Per i test, usare il seguente programma di prova `es2-main.c`:

```c
#include <stdio.h>

void update(int* v, int n);

void print_array(int* v, int n) {
    int i;
    printf("<");
    for (i=0; i<n; i++) printf("%s%d", i==0 ? "" : ", ", v[i]);
    printf(">");
}

int main() {
    int v1[] = { 10, 20, 30, 40 };
    int v2[] = { 13 };
    int v3[] = { };

    update(v1, 4);
    print_array(v1, 4);
    printf(" [corretto = <40, 30, 20, 10>]\n");

    update(v2, 1);
    print_array(v2, 1);
    printf(" [corretto = <37>]\n");

    update(v3, 0);
    print_array(v3, 0);
    printf(" [corretto = <>]\n");

    return 0;
}
```

Generare un file eseguibile `es2` compilato con `gcc –m32`.

Esercizio 3 (memoria virtuale)

1. Supponiamo di avere uno spazio logico di 4 GB suddiviso in pagine di 4 KB. Quanti bit occupa l'offset all'interno della pagina e quanti bit occupa il numero di pagina in un indirizzo di memoria logico?
2. Si supponga di avere una tabella delle pagine con celle a 32 bit che occupa 64 MB. Se il sistema di memoria usa pagine da 8 KB, quanto è grande lo spazio virtuale? Mostrare i calcoli effettuati.
3. Si consideri il seguente programma C eseguito su una piattaforma con memoria virtuale gestita con pagine da 4 KB:

```c
short x[1024];
int main() {
    int y[4096];
    double* z = malloc(1000000);
    return 0;
}
```

Quante pagine occupano, approssimativamente, le sezioni DATA, HEAP e STACK del programma subito prima dell'istruzione `return`? Si assuma che `short` occupi 2 byte, `int` 4 byte e `double` 8 byte.

Inserire le risposte nel file `es3.txt`.

Esercizio 4 (ottimizzazione di programmi)

Si crei nel file `es4-opt.c` una versione ottimizzata manualmente del seguente modulo `es4.c`, dove la funzione `sum_range` calcola la somma degli elementi di una lista collegata con indici compresi tra `a` incluso e `b` escluso (il primo nodo ha indice zero):

```
#include "es4.h"

nodo* get(nodo* p, int i) {
    for (; p != NULL && i--; p = p->next);
    return p;
}

int sum_range(nodo* list, int a, int b) {
    int i, s = 0;
    for (i=a; i<b; i++)
        s += get(list, i)->info;
    return s;
}
```

Compilare due versioni del programma, **usando `gcc` a 32 bit con livello di ottimizzazione O1** e lo stesso modulo `es4-main.c`:

1. Non ottimizzata: eseguibile `es4`.
2. Ottimizzata: eseguibile `es4-opt`.

Ai fini dell'ottimizzazione:

1. Usare `gprof` per identificare le porzioni più onerose computazionalmente. Chiamare gli eseguibili usati per la profilazione `es4-pg` e `es4-opt-pg`. Salvare i report di `gprof` nei file `es4.txt` e `es4-opt.txt`, rispettivamente
2. Esaminare il modulo assembly `es4.s` fornito (generato a partire da `es4.c` con `gcc -m32 -S -O1`) per capire quali ottimizzazioni siano già state effettuate dal compilatore.

Rispondere alle seguenti domande:

1. Descrivere le ottimizzazioni applicate e dire perché si ritiene che siano efficaci.
2. Riportare il tempo di esecuzione di `es4` e di `es4-opt` usando il comando `time`.
3. Riportare i flat profile delle due versioni usando `gprof`.
4. Di quante volte è più veloce l'eseguibile `es4-opt` rispetto a `es4`?
5. Usando i dati del profilo `es4.txt`, calcolare lo speedup massimo che si può ottenere ottimizzando la funzione `sum`. Si tenga presente che `gprof` misura i tempi in modo approssimato con una precisione al centesimo di secondo.

Inserire le risposte nel file `es4.txt`. Alla fine del compito, **non eliminare i seguenti file**:

- `es4`
- `es4-pg`
- `es4.txt`
- `es4-opt`
- `es4-opt-pg`
- `es4-opt.txt`
- `gmon.out`