Sistemi di Calcolo (A.A. 2016-2017)
Corso di Laurea in Ingegneria Informatica e Automatica
Sapienza Università di Roma

A

Compito di esonero – Durata 1h 30'

Inserire nome, cognome e matricola nel file `studente.txt`.

Parte 1 (programmazione IA32)

Si traduca in assembly IA32 la seguente funzione C scrivendo un modulo `es1A.s`:

```
int media(int, int);
int test(const short* x, const short* y) {
    return *x < media(*x,*y) && media(*x,*y) < *y;
}
```

L'unico criterio di valutazione è la correttezza, cioè l'equivalenza semantica tra il programma tradotto e quello C di partenza. Per i test, usare il programma di prova `es1A-main.c` e il modulo `es1A-media.s`, entrambi inclusi.

Generare un file eseguibile `es1A` compilato con `gcc –m32`.

Parte 2 (programmazione IA32)

Si traduca in assembly IA32 la seguente funzione C scrivendo un modulo `es2A.s`:

```
void concatReverse(char* s1, char* s2, char* dest,
                   int len_s1, int len_s2) {
    int i = 0;
    s1 = s1 + len_s1 - 1;
    for (; i < len_s1; i++) dest[i] = *s1--;
    while (*s2) dest[i++] = *s2++;
    dest[i] = 0;
}
```

L'unico criterio di valutazione è la correttezza, cioè l'equivalenza semantica tra il programma tradotto e quello C di partenza. Per i test, usare il programma di prova `es2A-main.c` incluso.

Generare un file eseguibile `es2A` compilato con `gcc –m32`.

Parte 3 (ottimizzazione work)

Si crei nel file `pila-opt.c` una versione **ottimizzata** del seguente modulo `pila.c`:

```
#include <stdlib.h>
#include "pila.h"

typedef struct nodo nodo;

struct nodo { // nodo lista
    int   elem;
    nodo* next;
};

struct pila {
    nodo* top; // nodo top della lista
};

pila* pila_new(){ // crea pila vuota
    return calloc(1, sizeof(pila));
```

```
}

int pila_len(const pila* p){ // lunghezza
    nodo* n;
    int c = 0;
    for (n = p->top; n != NULL; n = n->next) c++;
    return c;
}

void pila_push(pila* p, int x){ // push in cima
    nodo* n = malloc(sizeof(nodo));
    n->elem = x;
    n->next = p->top;
    p->top  = n;
}

int pila_pop(pila* p, int* x){ // pop dalla cima
    if (pila_len(p) == 0) return -1;
    nodo* dead = p->top;
    if (x != NULL) *x = dead->elem;
    p->top = dead->next;
    free(dead);
    return 0;
}

void pila_del(pila* p){ // dealloca pila
    while (p->top != NULL) pila_pop(p, NULL);
    free(p);
}
```

Il modulo implementa un **tipo di dato pila utilizzando una lista collegata**.

Compilare due versioni del programma, usando `gcc` **a 32 bit** con livello di ottimizzazione 1 e lo stesso modulo `main.c`:

1. non ottimizzata manualmente: eseguibile `pila`;
2. ottimizzata manualmente: eseguibile `pila-opt`.

Ai fini dell'ottimizzazione:

1. usare `gprof` per identificare le porzioni più onerose computazionalmente. Per evitare confusione, chiamare l'eseguibile usato per la profilazione `pila-pg` e il report del profiler `pila-pg.txt`;
2. esaminare il modulo `pila.s` generato a partire da `pila.c` con `gcc –S -O1` (già fornito) per capire quali ottimizzazioni siano già state effettuate dal compilatore.

Rispondere alle seguenti domande:

1. Descrivere le ottimizzazioni applicate **manualmente** e dire perché si ritiene che siano efficaci.
2. Riportare la media dei tempi di esecuzione (real) di `pila` e di `pila-opt` su tre run usando il comando `time` e dire di quante volte è più veloce l'**eseguibile** `pila-opt` rispetto a `pila` (speedup).
3. Riportare il flat profile del programma `pila` usando `gprof`.

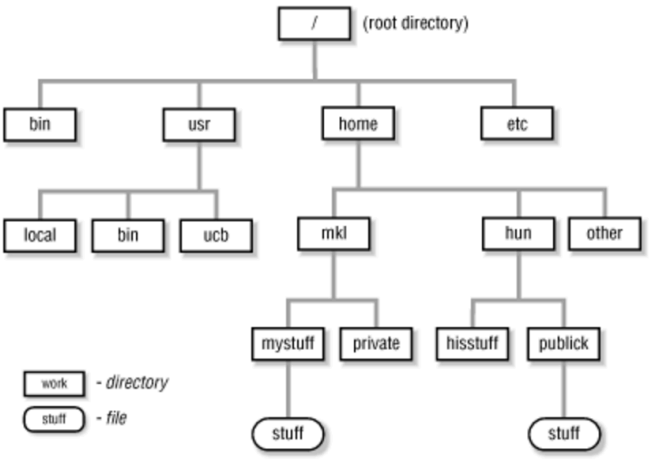
Inserire le risposte nel file `es3A.txt`. Alla fine del compito, **non cancellare** `gmon.out` e gli altri eseguibili creati.

Parte 4 (quiz)

Si risponda ai seguenti quiz, inserendo le risposte (A, B, C, D o E per ogni domanda) nel file `es4A.txt`. **Una sola risposta è quella giusta**. Rispondere E equivale a non rispondere (0 punti).

Domanda 1 (file system)

Si consideri la seguente struttura di directory:



Assumendo che la directory corrente sia `mkl`, quale comando occorre dare per eliminare la directory `publick`?

A	`delete ../hun/./publick`	**B**	`rm -rf /home/hun/./.publick`
C	`rm ../hun/./.publick`	**D**	`rmdir /home/hun/./.publick`

Motivare la risposta nel file `M1.txt`. **Risposte non motivate saranno considerate nulle.**

Domanda 2 (parametri main)

Se un programma C viene compilato (producendo il binario `prog`) ed eseguito utilizzando il comando `./prog uno 1`, quale delle seguenti espressioni booleane risulta vera nel momento in cui inizia l'esecuzione della sua funziona main:

A	`argc < 3`	**B**	`strcmp(argv[0], "uno") == 0`
C	`strcmp(argv[2], "1") == 0`	**D**	`argv[2] == 1`

Motivare la risposta nel file `M2.txt`. **Risposte non motivate saranno considerate nulle.**

Domanda 3 (analisi delle prestazioni del software)

Dato un programma con tre funzioni A, B e C, esse occupano rispettivamente il 40%, 40% e 20% del tempo di esecuzione. Sapendo che C è invocata soltanto da B, il compilatore può fare inlining di C in B ed ottimizzare la nuova versione di B. Qual è lo speedup massimo ottenibile per il nuovo programma se A rimane invariata?

A	2.50x	**B**	2.00x
C	2.25x	**D**	2.75x

Motivare la risposta nel file `M3.txt`. **Risposte non motivate saranno considerate nulle.**