

# Prova pratica di Calcolatori Elettronici

*C.d.L. in Ingegneria Informatica, Ordinamento DM 270*

18 settembre 2017

1. Siano date le seguenti dichiarazioni, contenute nel file `cc.h`:

```
struct st1 { char vc[4]; }; struct st2 { int vd[4]; };
class cl
{
    st1 s; long v[4];
public:
    cl(char c, st2& s2);
    void elab1(st1 s1, st2 s2);
    void stampa()
    {
        int i;
        for (i=0;i<4;i++) cout << s.vc[i] << ' '; cout << endl;
        for (i=0;i<4;i++) cout << v[i] << ' '; cout << endl << endl;
    }
};
```

Realizzare in Assembler GCC le funzioni membro seguenti.

```
cl::cl(char c, st2& s2) {
    for (int i = 0; i < 4; i++) {
        s.vc[i] = c + i;
        v[i] = s2.vd[i] + s.vc[i];
    }
}
void cl::elab1(st1 s1, st2 s2) {
    cl cla('a', s2);
    for (int i = 0; i < 4; i++) {
        if (s.vc[i] <= s1.vc[i]) {
            s.vc[i] = i + cla.s.vc[i];
            v[i] = i - cla.v[i];
        }
    }
}
```

2. Vogliamo aggiungere al nucleo un meccanismo tramite il quale un processo sistema può temporaneamente trasformarsi in un processo utente, eseguendo una funzione del modulo utente e riprendendo il controllo quando questa termina (tramite `terminate_p()` o, in caso di errore, `abort_p()`).

Per farlo aggiungiamo la seguente primitiva (invocabile solo da livello sistema):

```
bool call_user(addr f, natq regs[N_REG_GEN], natq *stack, natl n);
```

La primitiva deve saltare a **f** dopo aver riportato il processore a livello utente. Dopo il salto, il contenuto dei registri generali deve essere quello contenuto nell'array **regs** e la pila utente deve contenere le parole quaduple **stack[0]** (in cima) ... **stack[n-1]** (in fondo). Quando la funzione **f** termina (invocando **terminate\_p()** o **abort\_p()**) l'esecuzione deve tornare al chiamante della **call\_user()**, nuovamente a livello sistema; l'array **regs** deve ora contenere i valori dei registri generali al momento della terminazione della funzione **f**. Infine, la primitiva **call\_user()** deve restituire **false** se la trasformazione in processo utente non è andata a buon fine (per esempio, non è stato possibile creare la pila utente) e **true** altrimenti.

Per realizzare la primitiva aggiungiamo i seguenti campi al descrittore di processo:

```
natq contesto_salvato[N_REG_GEN];
natq pila_salvata[5];
natq *regs;
```

Dove **contesto\_salvato** serve a memorizzare lo stato dei registri generali (corrispondenti ai primi **N\_REG\_GEN** campi del campo **contesto**) prima di scrivervi il contenuto dell'array **regs** (in modo da poterli ripristinare alla terminazione della funzione utente); il campo **regs** contiene un puntatore all'array omonimo passato alla **call\_user**; si veda sotto per il campo **pila\_salvata**.

Si noti che la **call\_user()**, come tutte le primitive, sarà invocata tramite una istruzione **INT**, la quale salverà in pila sistema le 5 parole che una successiva **IRETQ** potrà estrarre per tornare all'istruzione successiva alla **INT** stessa. Un puntatore alla prima (dall'alto) di queste parole è contenuto nel campo **contesto[I\_RSP]** del descrittore di ogni processo.

Per svolgere il suo compito, la **call\_user()** deve modificare opportunamente le 5 parole lunghe salvate dalla **INT** che l'ha messa in esecuzione, e quindi terminare. Questo deve produrre il salto a livello utente nello stato descritto precedentemente. Le precedenti parole lunghe devono essere memorizzate nel descrittore di processo (nel nuovo campo **pila\_salvata**). La **terminate\_p()** e la **abort\_p()**, quando trovano un valore non-nullo nel campo **regs** del descrittore del processo che le ha invocate, capiscono che questo processo aveva precedentemetne invocato una **call\_user()** e, invece di distruggerlo, ripristinano il contesto e la pila salvati, in modo da ritornare al chiamante della **call\_user()** stessa.

Modificare il file **sistema.cpp** per aggiungere le parti mancanti nella realizzazione di questo meccanismo.

**ATTENZIONE:** i processi sistema non hanno pila utente e hanno un campo **punt\_nucleo** nullo. Per poter trasformare il processo sistema in utente, la **call\_user()** dovrà anche creare la pila utente e inizializzare **punt\_nucleo**.

**SUGGERIMENTO:** si tenga presente che la parte di pila sistema che si trova sotto le 5 parole può contenere informazioni utili per il processo sistema che ha invocato la **call\_user()**, e dunque non deve essere sovrascritta mentre è in esecuzione la funzione utente.