Domande orale 22/07

giovedì 22 luglio 2021 09:30

Ad inizio orale il prof mostra al candidato la prova pratica svolta e se ha qualche dubbio a riguardo può chiedere chiarimenti, quindi è consigliato imparare la teoria dietro agli errori commessi durante la prova pratica e la loro soluzione

Corrispondenza = il prof ti detta una o piu strutture e/o una classe, una o due funzioni e bisogna tradurle
In assembler. Bisogna spiegargli cosa stiamo facendo mentre svolgiamo l'esercizio.
L'esercizio viene fatto condividendo il proprio schermo e svolgendo l'esercizio su un editor a scelta.

Nota bene: le soluzioni degli esercizi di traduzione non sono affidabili al 100%, in quanto il prof potrebbe non essersi curato di eventuali errori o comunque potrebbe aver fermato lo studente prima dal completamento dell'esercizio, quindi si raccomanda di non farci affidamento ma di prenderle come esempio di svolgimento iniziale dell'esercizio. Eventualmente consiglio di scrivere l'esercizio con un main semplice e una funzione di stampa e provare a vedere se la traduzione raggiunta si comporta come sperato.

1)Corrispondenza

```
struct s{
    char c;
    int a[2];
};

void g(int* p);

int f(s s1){
    g(&s1.a[1]);
    return s1.a[1];
}
```

Soluzione

```
_Z1f1s:
    push %rbp
    mov %rsp, %rbp
    sub $16, %rsp

    mov %rdi, -16(%rbp)
    mov %esi, -8(%rbp)

    lea -8(%rbp), %rdi

    call _Z1gPi

    mov -8(%rbp), %eax

    leave
    ret
```

2) Ottimizzazione che prevede di non mettere i dati in memoria e tenerli nei registri(red zone)
   No perché bisogna passare il puntatore a g
3) in quali problemi andiamo incontro con un operazione di DMA in presenza della cache?
problema di consistenza della ram, la cache fa write-back e quindi una eventuale lettura potrebbe darmi dati non aggiornati.
il ponte scrive in ram e la cache successivamente riscrive nella zona di memoria in cui ha scritto il ponte.
Se il ponte mi aggiorna la ram, il software vuole leggere ma la cache mi da gia i dati non aggiornati.
Soluzione: hold, holda e quando il ponte scrive in ram scrive anche in cache.

Invalidamo la cacheline tramite i bit dirty in caso il ponte scrivesse in ram se la cache è write through

4) Corrispondenza

```
struct s{
    char c;
    int a[2];

};

struct s1{
    short w;
    s t;
};

s1 f(int i){
    s1 s;
    s.t.a[0] = i;
    return s;
}
```

Soluzione

```
.text
.global _Z1fi
_Z1fi:

PUSH %RBR
MOV %RSP, %RBP
SUB $32, %RSP

MOV %EDI, -8(%RBP)

MOV %EDI, -16(%RBP)

MOV -24(%RBP), %RAX
MOV -16(%RBP), %RDX

LEAVE
RET
```

5) Stessa domanda 2: l'ottimizzazione era possibile perché non dovevamo passare nessun parametro e quindi potevamo lasciare tutto nei registri +
6) Perché siamo passati dalla soluzione driver a quella dell'handler + processo esterno?
   Perché avere le interruzioni disabilitate è una grossa limitazione, il driver non è un processo quindi non possiamo interromperlo

7) problemi di programmazione con il processo esterno che non abbiamo con il driver?
quando accediamo alle strutture dati del sistema bisogna usare primitive del sistema per evitare di lasciarle inconsistenti

8) Che operazioni deve eseguire la carica_stato? Nel modo più dettagliato possibile(ragazzo a cui è stato conservato lo scritto dopo non aver passato l'orale l'appello precedente)
   È richiesto di scrivere la carica_stato.

9) quali controlli fa l'iretq?
Controlla cpl e si assicura che sia quello salvato in pila

10) Perché cambiamo pila?
    Lo facciamo quando c'e cambio di livello e si passa a livello sistema , per evitare che l'utente possa fare delle istruzioni ad hoc e possa innalzare il proprio livello
11) Perché usiamo una pila sistema diversa per ogni processo?
    Se ci fosse solo il modulo sistema ne basterebbe una, essendo atomico la sporcherebbe e la pulirebbe prima che un altro processo la possa utilizzare però avendo anche il modulo io , i processi si posso bloccare quindi abbiamo bisogno che lo stato del processo sia salvato in una pila sicura non accessibile all'utente
12) Corrispondenza

```
class cc
{
    long a[4];
    cc(const cc& c1);
    cc f()
    {
        cc tmp();
        for (int i = 0; i < 4; i++)
        {
            tmp.a[i]=i;
        }
        return tmp;
    }
};
```

13) Come realizziamo le primitive? Partendo dallo spazio utente, assembler e c++
Come inizializziamo i gate della IDT?

14) Corrispondenza

```
class cc{

    long array[4];

    cc();
    cc(int a);

    cc f(){
        cc tmp(3);
        for (int i=0; i<4; i++){
            tmp.array[i] +=  array[i];
        }
        return tmp;
    }
}
```

15) Consideriamo un handler, quando va in esecuzione? Dopo che l'apic manda l'interruzione associata all'handler
L'handler gira a interruzioni abilitate o no? No perche si trova nel modulo sistema quindi è atomico
La iretq dell'handler su quale pila sistema agisce? Quella del processo esterno dato che esecuzione punta al processo esterno.
Che stato c'e salvato nella pila sistema del processo esterno? Dove salterà la iretq?
Possiamo essere sicuri che ripartiremo dalla fine della wfi, anche se un processo ad alta priorità volesse andare in esecuzione comunque cambio la variabile esecuzione solo quando facciamo la wfi e quindi abbiamo il cambio di processo perché la wfi sospende il processo e chiama schedulatore.