

Domande orale 01-07

giovedì 1 luglio 2021 09:19

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 più 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 c1;  
    long l;  
    char c2;  
};  
  
s fun(s* mias){  
    return *mias;  
}
```

cambio di domanda

```
s fun(s* mias) diventa ---->>>  
---->> s fun() {  
    s temp;  
    return temp;  
}
```

2) come funzionano i byte enable, come è organizzata la memoria per ottenere i byte enable (vuole il disegno)

Risposta: sono 8 e servono

3) perché quando scriviamo le primitive bisogna passare dall'assembler? Sia dalla parte sistema sia dalla parte utente

4) Corrispondenza: stessa struttura di 1)

Soluzione:

```

.global: _Z3funPss
_Z3funPss:
    push %rbp
    mov %rsp,%rbp
    sub $16,%rsp

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

    mov $3,%rcx
    rep movsq
    mov -8(%rbp),%rax
    leave
    ret

```

5) S fun() {
 s temp;
 return temp;
}

```
S fun() {
    return s();
}
```

Differenza tra queste due funzioni ?

Nella seconda viene applicata la RVO, lavoro direttamente sull'indirizzo di ritorno
 Nella prima si potrebbe effettuare la RNVO, dove non allochiamo l'oggetto ma lavoriamo sull'indirizzo di ritorno

6) Corrispondenza:

```

struct s {
    int i1, i2;
};

struct s2 {
    char c1;
    s mias;
};

void g(s*);

void f(s2 mias2) {
    g(&s2.s);
}

```

Soluzione:

```

.global _Z1f2s2
_Z1f2s2:
    pushq %rbp
    movq %rsp, %rbp
    subq $16, $rsp

    # copia parametri
    movq %rdi, -16(%rbp)
    movl %esi, -8(%rbp)

    leaq -12(%rbp), %rdi
    call _Z1gP1s

    leave
    ret

```

- 7) Avremmo potuto evitare di copiare i parametri nei registri?

No, perché devo passare il puntatore a s e quindi devo mettere i parametri in pila

- 8) Tlb: cosa indica il bit D salvato nel tlb, come viene salvato.

La mmu se trova il bit D a 0 nel tlb, causa una miss e quindi ripercorre l'albero di traduzione.

Non possiamo risolvere il problema via software perché il tlb è trasparente al software e comunque il tlb non ha informazioni sufficienti per poter settare un bit D a piacere

- 9) corrispondenza:

```

class cc{
    long v[4];

    cc(long*);

    void f(int i, long* l){
        cc temp(l);
        v[i] += temp.v[i]
    }
}

```

soluzione:

```

.global _Z2cc1fEiPl
_Z2cc1fEiPl:
    push %rbp
    mov %rsp, %rbp
    sub $64, %rsp

    mov %rdi, -56(%rbp)
    mov %esi, -8(%rbp)
    mov %rdx, -16(%rbp)

    lea -48(%rbp), %rdi
    mov -16(%rbp), %rsi
    call _Z2ccC1EP1

    movslq -8(%rbp), %rdx
    mov -48(%rbp), %rdx, 8, %rax
    mov -56(%rbp), %rsi
    mov %rax, (%rsi, %rdx, 8)

    leave
    ret

```

- 10) Accesso in memoria di tlb che causa una miss, quanti accessi in memoria devo fare nel caso peggiore? Se devo percorrere tutto l'albero, la mmu deve fare altre cose oltre a tradurre? Facendo l'ipotesi che l'indirizzo non sia nel tlb.
 Ci sono altri bit oltre a P quindi deve controllare questi bit. BIT: pcd, Pwd, us, Rw, D, A.
 Quindi quando la mmu passa da una entrata deve settare il bit A e se l'accesso è in scrittura anche il bit D
 Il numero di accessi in lettura e in scrittura: un accesso per leggere sia il numero di pagina che i bit e un altro eventuale accesso in scrittura per aggiornare i bit. Quindi in totale abbiamo 8 accessi in memoria nel caso peggiore.

- 17) Corrispondenza:

```
class cc {
    long v[4];

    void f(cc &c1) {
        cc c2;
        for(int i=0; i<4; i++) {
            c2.v[i] = v[i] + c1.v[i];
        }
    }
}
```

Soluzione:

```
_Z3cc1ers:
    push %rbp
    mov %rsp, %rbp
    sub $64, %rsp

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

    movl $0, -56(%rbp)
.Lfor:
    cmpl $4, -56(%rbp)
    jl .Lcorporfor
    jmp .Lfinefor
.Lcorporfor:
    movslq -56(%rbp), %rcx
    mov (%rsi, %rcx, 8), %rax
    mov (%rdi, %rcx, 8), %rdx
    add %rdx, %rax
    mov %rax, -48(%rbp, %rcx, 8)
    incl -56(%rbp)
    jmp .Lfor
.Lfinefor:
    leave
    ret
```

- 18) Quali azioni esegue un handler? (scrivere codice)

Handler_i:

```
call salva_stato
call inspronti
Movq $i, %rcx
Movq a_p(%rcx, 8), %rax
Movq %rax, esecuzione
Call carica_stato
Iretq
```

- 19) Corrispondenza:

```

class cc{
    long l[4];

    void f(cc* mioc) {
        if(mioc->l[2] > l[2])
            l[2] = mioc->l[2];
        else
            l[2] = mioc->l[3];
    }
}

```

Soluzione parziale:

```

# this
MOV %RDI, -8(%RSP)

# mio
MOV %RSI, -16(%RSP)

# if(mioc->l[2] > l[2])
MOV 16(%RDI), %RAX

CMP %RAX, 16(%RSI)
JG then
JMP else

then:
    MOV 16(%RSI), %RAX
    MOV %RAX, 16(%RDI)
    JMP endif

else:
    MOV 24(%RSI), %RAX
    MOV %RAX, 16(%RDI)

endif:
    LEAVE
    RET

```

- 20) Handler: codice e a cosa serve
- 21) A quale istruzione si salterà alla fine della iretq dell'handler? In che punto del processo esterno ci troveremo? Dopo aver svolto la wfi(), cioe all'inizio del corpo del for oppure all'inizio del processo esterno se non è mai andato in esecuzione
- 22) Quale comportamento dell'apic ci garantisce che l'handler di un tipo non rivada in esecuzione
- 23) Se la primitiva del compito prevedesse la sincronizzazione in un cui un processo cede il controllo e si blocca, sarebbe stato necessario invalidare il tlb ? No perche il processo cambia e il tlb verrebbe invalidato automaticamente
- 24) Corrispondenza:

```

struct s{ # allineamento 4 dim
    int a[3];
}
struct s2{
    s b[3];
}
void f(s2& s){
    for(i=0; i<3; i++){
        s.b[i].a[i]=i;
    }
}

```

Soluzione:

```
.global _Z1fR2s2
```

```

.global _Z1fR2s2
_Z1fR2s2:
    push %rbp
    mov %rsp, %rbp
    sub $16, %rsp

    mov %rdi, -8(%rbp)
    movl $0, -16(%rbp)

condFor:
    mov -16(%rbp), %ecx
    cmp $3, %ecx
    jb corpoFor
    jmp fineFor

corpoFor:
    # s.b[i].a[i]=i;
    mov -8(%rbp), %rsi
    mov %ecx, (%rsi)
    add $16, %rsi
    incl -16(%rbp)
    jmp condFor

fineFor:
    leave
    ret

```

- 25) L'apic fa passare le richieste a precedenza maggiore, spiegare come funziona questo meccanismo, come fa l'apic a sapere quali interruzioni gestire? Tramite i registri irr, isr
Il processore sposta il tipo da irr a isr solo se l'interruzioni sono abilitate
- 26) Come funzionano i due modi di riconoscimento? Cioe sul livello e sul fonte
Sul livello: fronte + se è ancora attiva la richiesta quando c'è EOI
Il timer

- 27) Corrispondenza

```

est.cpp > cc > l(cc)
class cc{
    char c1;
    short s[2];
    int i;

    cc f(cc c){
        c.s[0]=i;
        return c;
    }
};

```

Soluzione:

```

.global _ZN2cc1fES_
_ZN2cc1fES_:
    # prologo
    push %rbp
    mov %rsp, %rbp
    sub $32, %rsp

    # passaggio dati
    mov %rdi, -8(%rbp)
    mov %rsi, -24(%rbp)
    mov %edx, -16(%rbp)

    mov 8(%rdi), %eax
    mov %eax, -22(%rbp)

    mov -24(%rbp), %rax
    mov -16(%rbp), %edx
    leave
    ret

```

- 28) Operazione di DMA, l'utente usa una primitiva e ci passa un buffer, abbiamo la memoria virtuale, a cosa bisogna stare attenti?

L'indirizzo del buff è virtuale ma un operazione in dma non passa dalla mmu e quindi va passato il suo indirizzo fisico.

Il buff in memoria fisica potrebbe essere non contiguo e appartenere a frame diversi per evitare ciò dobbiamo fare più trasferimenti e ogni volta controllare che il numero di dati che inviamo non deve far sì che l'indirizzo del buffer vada al di là del frame di appartenenza(vedere le soluzioni delle prove d'esame degli es sul nucleo che riguardano la dma)

- 29) Corrispondenza

```

#C++
struct s {
    int i;
    s * next;
};

s* f(s* head){
    s* tmp = head;
    if(head)
        head = head->next;
    return tmp;
}

```

Soluzione:

```

.global _Z1fP1s
_Z1fP1s:
    push %rbp
    mov %rsp, %rbp
    sub $16, %rsp
    mov %rdi, -8(%rbp)
    mov %rdi, -16(%rbp)
    cmp $0, %rdi
    jne .Lfineif
    mov 8(%rdi), %rax
    mov %rax, -8(%rbp)
.Lfineif:
    mov -16(%rbp), %rax
    leave
    ret

```

- 30) Supponiamo di non volere la finestra sulla memoria nella memoria virtuale di ogni processo
 Ma alcune strutture dati devono poter essere accessibili, quali sono queste strutture dati?
 Le strutture dati che devo avere necessariamente mappate sono:

La idt

La gdt -> tss

La pila sistema

Della ruotine di sistema mi serve solo quanto basta per cambiare la tabella di traduzione

- 31) Corrispondenza

```

struct s1{
    char c[3];
    int i;
};

struct s2{
    char cl;
    s1 s;
};

int f(s2 s)
{
    return s.s.i;
}

```

è richiesta la traduzione di f

- 32) Supponiamo che un utente voglia fare un'operazione di io e passa alla primitiva una buffer,
 Che vincoli ci sono sul buffer dell'utente?
 Ora al contrario di una operazione di DMA si passa dalla MMU quindi i vincoli sono quelli
 controllati dalla Access.
 33) se invece l'operazione fosse in bus mastering? Che vincoli ci sono sul buffer? Vedere
 risposta domanda 28
- 33) In che occasioni bisogna invalidare il tlb?
- 34) Corrispondenza:

```

class cc{
    int a[4];

    cc(const cc &c1){
        a[0] = c1.a[0] + 1;
    }

    cc f(cc &c2){
        cc tmp(c2);
        tmp.a[0]++;
        return tmp;
    }
}

```

Solo di f

Soluzione:

Serve anche l'indirizzo del risultato passato dal chiamante perché c'e il costruttore di copia ridefinito

+-----+			
a[1]	a[0]		
+-----+			
a[3]	a[2]		
+-----+			
+-----+			
a[1]	a[0]	-40	
+-----+			
a[3]	a[2]	-32	
+-----+			
c2		-24	
+-----+			
this		-16	
+-----+			
ind risu		-8	
+-----+			
old rbp		<- rbp	
+-----+			
ind rit			
+-----+			

```

.global _ZN2cc1fERS_
_ZN2cc1fERS_:
    push %rbp
    mov %rsp, %rbp
    sub $48, %rsp

    mov %rdi, -8(%rbp)
    mov %rsi, -16(%rbp)
    mov %rdx, -24(%rbp)

    lea -40(%rbp), %rdi
    mov -24(%rbp), %rsi
    call _ZN2ccC1ERKS_

    incl -40(%rbp)

    mov -8(%rbp), %rdi
    lea -40(%rbp), %rsi

    leave
    ret

```

- 35) A cosa servono i byte enable? Come sono collegati? (chiede il disegno)
 37) perché quando scriviamo le primitive siamo costretti a scriverne una parte in assembler?
 In c++ non possiamo usare iretq, int e non potremmo realizzare salva_stato e carica_stato visto che usano i registri del processore.

- 36) Corrispondenza

```

class cc
{
    long v[8];
    cc(int);
    void f(cc& c1)
    {
        cc tmp{c1.v[0]};
        for(int i = 0; i < 8; i++)
        {
            v[i] = tmp.v[i] + c1.v[8-i];
        }
    }
};

```

Soluzione: non corretta in parte

```

.global _ZN2cc1fERS_
_ZN2cc1fERS_:
    push %rbp
    mov %rsp, %rbp

    sub $(12*8), %rsp

    mov %rdi, -24(%rbp)
    mov %rsi, -32(%rbp)

    lea -96(%rbp), %rdi
    mov (%rsi), %esi
    call _ZN2ccC1Ei

    mov $8, %rcx
.Lfor:
    mov %ecx, -16(%rbp)

    mov (%rax,%rcx,8), %r8 #tmp.v[i]
    mov $8, %r9d
    sub %ecx, %r9d

    mov -32(%rbp), %rdx
    mov (%rdx, %r9, 8), %rsi #c1.v[8-i]

    mov -24(%rbp), %rdi

    add %r8, %rsi|
    mov %r15, (%rdi, %rcx, 8)

    loop .Lfor

    leave
    ret

```

- 37) Che problemi ci sono tra bus mastering e cache?

Dobbiamo mantenere la consistenza tra cache e ram, il ponte scrive in ram e la cache contiene quindi dati vecchi, oppure se abbiamo una cache con politica write-back quando il bus va a leggere troverebbe dati che non sono stati ancora aggiornati dalla cache.

Nel caso il ponte non scrivesse o leggesse in cacheline i problemi sarebbero di più, se scrivesse un singolo byte

sarebbe più complicato, bisogna fare in modo che se una cacheline ha un bit dirty la cache deve dare una miss.

Il ponte ospite/pci deve comunicare con la cache prima di comunicare con la cache.

Tramite software: possiamo levare la politica write-back della cache tramite il bios e usare la operazione per invalidare la cache dopo che l'operazione di bus mastering si è conclusa.

- 38) Corrispondenza

```

struct s{
    int i;
    char c1, c2;
};

struct s1{
    s mias;
    short w;
};

void f(s1 &ss){
    ss.w = ss.mias.c1 + ss.mias.c2;
}

```

- 39) Perché quando attraversiamo la idt dopo avuto uno dei 3 casi (int , esterne, eccezioni) si cambia

la pila e si usa la pila sistema?

Perché vogliamo conservarci delle informazioni che l'utente non deve poter modificare in nessuno modo, se non la facessimo l'utente potrebbe usare una iretq ad hoc per aumentare il suo livello di privilegio tipo modificando ciò che è contenuto in CPL.

- 40) Il processore dove prende l'indirizzo di questa nuova pila?

Lo prende dal tss, c'è un registro tr che contiene ad ogni momento il puntatore all'entrata del tss che contiene l'indirizzo della pila sistema corrente.

Chi inizializza questo tss? Ogni volta che c'e il cambio di processo il tss deve essere inizializzato con l'indirizzo della pila corrente

Perché il processore salva il registro dei flag quando attraversa l'idt?

- 41)

```
#include <cs.h>
1 struct s{
2     short s1;
3     short s2;
4 }
5
6 int f(s a[], int i ){
7
8     int w = 0;
9     for (int j = 0 ; j < i ; j++){
10         w += a[j].s2;
11     }
12
13     return w;
14 }
```

Soluzione:

- 42) Quali strutture dati vanno inizializzate e come quando si crea un processo? (disegno)

- 43) Corrispondenza:

```
struct s{
    short s1, s2, s3;
};

int f(s a[], int m){
    int w = 1;
    for (i = 0; i < m; i++){
        w *= a[i].s2;
    }
    return w;
}
```

Soluzione:

```

.global _Z1fPisi
_Z1fPisi: push %rbp
    mov %rsp, %rbp
    sub $32, %rsp
    mov %rdi, -8(%rbp)
    mov %esi, -16(%rbp)
    movl $1, -12(%rbp)
    mov $6, %r8
.Lfor:   movl -16(%rbp), %edx
    cmp %edx, -24(%rbp)
    jge .Lfinelfor
    mov -12(%rbp), %eax
    mov -8(%rbp), %rcx
    mov (%rcx, %r8), %dx
    mul

```

- 44) Perché usiamo la paginazione? Che problemi ci risolve? Che vantaggi e svantaggi possiamo avere

nell'avere pagine più grandi o più piccole?

Con pagine più grandi il problema della frammentazione si risolve male però ha il vantaggio che ci fa risparmiare spazio in memoria dato dal dover allocare le tabelle di traduzione.

```

class cc{ // |
    int i;
    int *p;

    cc(int);

    void f(int a[], int j){
        cc tmp(j);
        for(int i = 0; i < j; i++){
            p[i] = a[i] + tmp.p[i];
        }
    }
};

```

- 45) Ci sono problemi di sincronizzazione tra un dispositivo e il ponte ospite/pci ?

Problema tra controllore-ATA, south bridge e ponte ospite/pci e come si risolve.

I dati possono essere rimasti nel north bridge e non sono arrivati in memoria, quindi come facciamo a sapere se i dati sono effettivamente arrivati? Il processore rischia di leggere dati vecchi.

- 46) Corrispondenza:

```

cpp.cpp

struct s1{
    char c[4];
};

class cc{
    s1 s;
    long l[4];

    cc f(){ //ZN2cc1fEv
        for (int i = 0; i < 4; i++) {
            l[i] += s.c[i];
        }

        return *this;
    }
}

```

Soluzioni:

```

pushq %rbp
movq %rsp, %rbp
subq %16, %rsp

movq %rdi, -8(%rbp)
movw %1, -24(%rbp)
movq -16(%rbp), %rdx    #l
movq -16(%rbp), %rax    #c
add %8, %rdx

.Lfor:
    cmp -24(%rbp), %4
    jge .LfineFor

.LcorpoFor:
    movw -24(%rbp), %rdi    #disp i
    movsbl (%rax), %r8
    addq %r8, (%rdx)

    add $1, %rax
    add $8, %rdx

    jmp .Lfor
.LfineFor:

    mov $5, %rcx
    mov -8(%rbp), %rdi
    mov -16(%rbp), %rsi

    rep movsq

    leave
    ret

```

- 47) A cosa serve la cache e come funziona?
- 48) Da cosa dipende il numero di bit dell'offset?
Dipendono dalla grandezza della pagina e solitamente sono 12 per le pagine grandi 4 kib e

21 per le pagine grandi 2 mib

47) che limitazioni hanno le cache ad accesso diretto? La cache non può caricare le cacheline dove gli pare e se più cacheline hanno lo stesso indice si genera una collisione anche se il numero di etichetta è diverso

49) Cache associative ad insieme, quali vantaggi offrono? Permettono di evitare collisioni tra cacheline che hanno lo stesso indice, hanno piu tabelle delle etichette.

50) a cosa servono i semafori?

Ci risolvono i problemi di muta esclusione e di sincronizzazione

In che modo ce li risolvono?