Corso di Laurea in Informatica, A.A. 2022/2023

Architetture degli Elaboratori e Sistemi Operativi (AESO corso A e B)

Terzo Appello – 4 Luglio 2023

Scrivere in modo comprensibile e chiaro rispondendo alle domande/esercizi riportati in questo foglio. Sviluppare le soluzioni dei primi due esercizi (prima parte) in un foglio separato rispetto alla soluzione degli ultimi due esercizi (seconda parte), in modo da facilitare la correzione da parte dei docenti. Riportare su tutti i fogli consegnati nome, cognome, numero di matricola e corso (A o B).

Parte 1

Esercizio 1

Si consideri una lista formata da elementi di tipo:

```
typedef struct __el { int v; struct __el *next; } ELEM;
```

e si assuma che sia a disposizione (ovvero esiste un'etichetta **fun** che ne denota il codice utilizzabile all'interno del programma) una funzione **fun** di firma:

```
int fun(int x);
```

Si fornisca il codice della funzione di firma:

```
void mapl(ELEM *p);
```

che sostituisce tutti i valori degli elementi della lista con il risultato del calcolo della funzione **fun** sul valore precedente dello stesso elemento. Se ad esempio **fun(x)** calcolasse semplicemente il successore del parametro intero, l'applicazione della **mapl** alla lista **1 → 2 → 3** la trasformerebbe nella lista **2→3→4** . Nel chiamare la funzione **fun** vanno rispettate tutte le solite convenzioni ARM sull'utilizzo dei registri.

Esercizio 2

Si progetti una rete sequenziale con ingressi IN a 2 bit, OP a 2 bit, e D a 32 bit. La rete mantiene internamente quattro registri per lo stato interno a 32 bit ciascuno chiamati R1, R2, R3 e R4. La rete produce un'uscita OUT a 32 bit. La specifica della rete è data dal seguente pseudocodice:

```
switch (OP)
  case 00: { R[IN] = R[IN] + D;
             OUT = R[IN] + D; }
  case 01: { OUT = R[IN] +D; }
  case 10: { OUT = R[IN] - D; }
  case 11: { R[IN] = R[IN] - D;
             OUT = R[IN] - D; }
```

Progettare la rete utilizzando esclusivamente componenti standard tra cui: full-adder con propagazione di riporto ad onda, multiplexer, demultiplexer e confrontatori. Definire la rete in termini di queste componenti standard e discutere una stima di un possibile lower bound alla durata del ciclo di clock in termini di numero di livelli di logica.

Parte 2

Esercizio 3

a) Descrive brevemente cosa si intende e quando si applica la tecnica Copy-on-Write (CoW).

b) Si consideri un sistema che gestisce la memoria con segmentazione paginata con le seguenti caratteristiche: indirizzi logici a 48 bit, pagine logiche e fisiche da 4 KB, numero massimo di segmenti per processo pari a 256, tabella delle pagine a 3 livelli dove la tabella di primo livello ha 1024 entry e le tabelle di secondo e terzo livello hanno lo stesso numero di entry. Ogni entrata di ogni tabella delle pagine ha ampiezza pari a 32 bit di cui 8 sono utilizzati per gli indicatori (P,R,W,U,…). In questo sistema è presente il processo *P* che ha allocato nello spazio virtuale un segmento *S* di dimensione 1.2 GB a partire dall'indirizzo logico (esadecimale) 0x200000000000. Si chiede:

1. Il formato dell'indirizzo logico indicando la lunghezza in bit delle componenti che indicizzano la tabella dei segmenti, le tabelle delle pagine ed il campo offset;
2. La dimensione in bit dell'indirizzo fisico e la massima dimensione della memoria fisica che un processo in esecuzione nel sistema può indirizzare;
3. Nell'ipotesi che tutte le pagine del segmento *S* siano state riferite, indicare il numero di tabelle di secondo livello necessarie per la traduzione degli indirizzi logici emessi dal processo *P*;
4. Indicare se l'indirizzo logico esadecimale 0x20004CCCD100 genera o meno una eccezione di violazione di memoria motivando la risposta.

Esercizio 4

a) Descrivere brevemente l'algoritmo MFQ con Affinity Scheduling per sistemi multiprocessore.

b) Una struttura dati è condivisa da 4 thread T1, T2, T3, T4. I thread T1 e T2 modificano la struttura condivisa, i thread T3 e T4 accedono alla struttura in sola lettura. Per l'accesso alla struttura dati si adotta la seguente politica:

a. Inizialmente accede il primo thread che ne fa richiesta sia esso lettore o scrittore;
b. Quando la struttura dati è in uso da T3 o da T4 (o da entrambi), se i thread T1 e T2 decidono di accedere alla struttura dati vengono messi in attesa. I thread in attesa vengono riattivati solo quando sia T3 sia T4 hanno completato l'accesso alla struttura dati. Se T1 e T2 sono entrambi in attesa allora viene riattivato uno dei due. Se né T1 né T2 sono in attesa allora la struttura dati ritorna disponibile a tutti i thread e si torna alla situazione iniziale;
c. Quando la struttura dati condivisa è utilizzata da T1 (o da T2), tutti gli altri thread che richiedono l'accesso vengono messi in attesa. Quando T1 (oppure T2) termina il proprio accesso, verifica prima se T2 (o T1) è in attesa e in caso affermativo lo riattiva. Altrimenti, se sono in attesa, vengono riattivati T3 e/o T4. Se nessun thread è in attesa allora la struttura dati ritorna disponibile a tutti i thread e si torna alla situazione iniziale.

Si chiede di implementare, utilizzando esclusivamente i semafori come meccanismo di sincronizzazione, i metodi: *accediT1T2(), rilasciaT1T2(), accediT3T4(), rilasciaT3T4()*.

SOLUZIONE

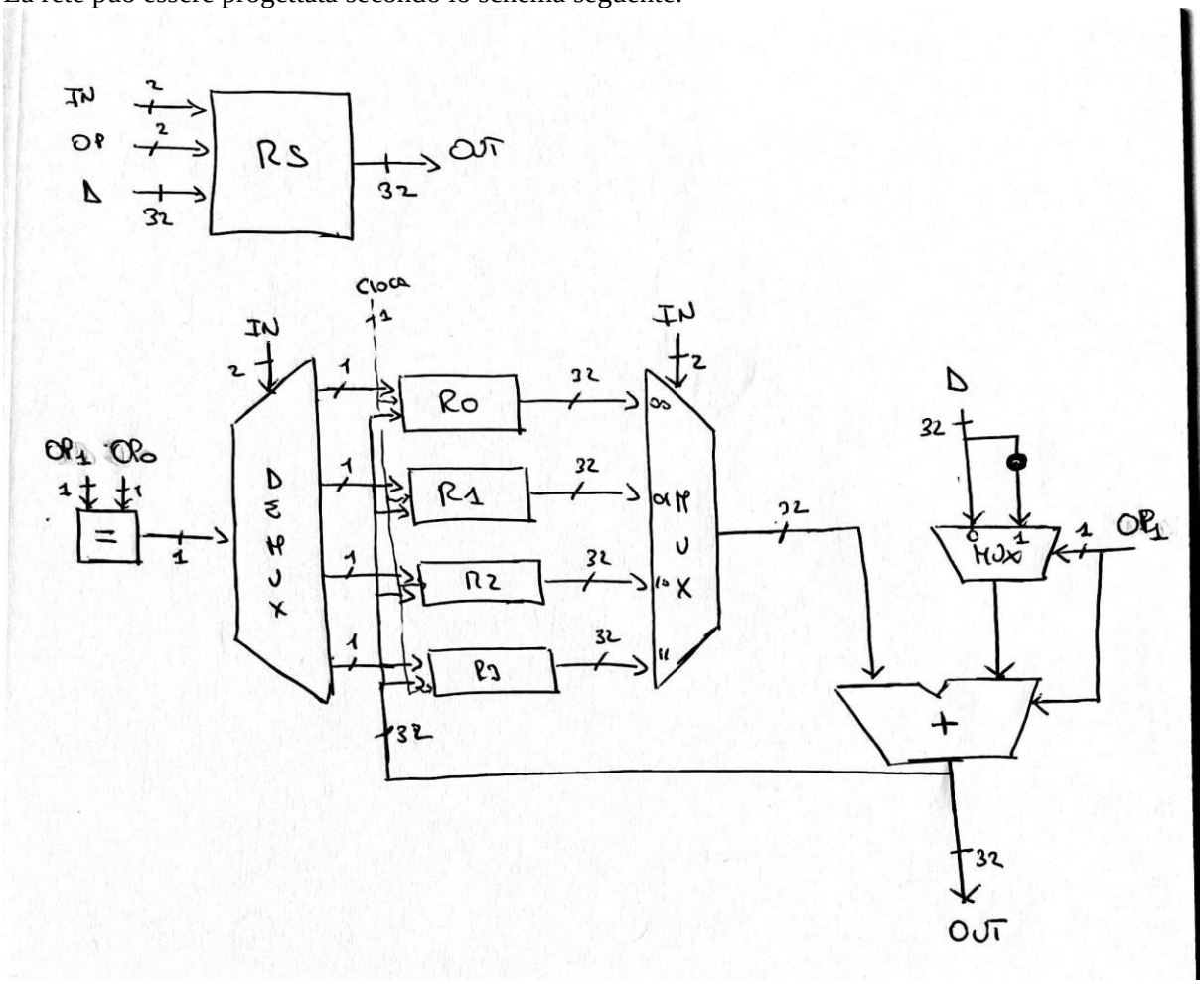
Eserizio 1

```
      .text
      .global mapl
      .type mapl, %function

mapl: cmp r0, #0       @ r0 puntatore alla lista
      moveq pc, lr     @ se null, ritorna (la somma e' gia' 0)
      push {r4, lr}    @ va salvato il parametro puntatore e lr
      mov r4, r0       @ r4 = puntatore all'elemento corrente
loop: ldr r0, [r4]     @ valore dell'elemento
      bl fun           @ chiama la funzione
      str r0, [r4]     @ cambia il valore attuale col risultato
      ldr r4, [r4,#4]  @ carica next
      cmp r4, #0       @ controlla se è null
      bne loop         @ se non è null, cicla
      pop {r4, pc}     @ altrimenti return
```

Esercizio 2

La rete può essere progettata secondo lo schema seguente:



Il confrontatore prende in ingresso i due bit di OP e fornisce l'ingresso al DEMUX che controlla il WE dei registri comandato da IN. In questo modo si scrive solo quando OP=00 oppure OP=11. La lettura avviene con un MUX comandato sempre da IN. L'operazione di sottrazione si realizza usando sempre il full-adder, andando a fornire però come resto in ingresso un bit uguale a 1 (si usa il bit più significativo di OP) e facendo il complemento a uno dell'ingresso D. Il piccolo MUX nel secondo ingresso del sommatore stabilisce se dobbiamo usare D oppure il suo complemento ad uno in caso di sottrazione (come abbiamo fatto nella ALU del processore).

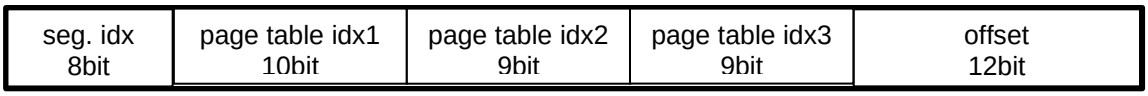
Un possibile lower bound al ciclo di clock si può stimare come:

$$\tau \geq max\{T \quad + T_{DEMUX}, T_{MUX} + T_{ADDER}\}$$

Si osserva che il confrontatore richiede due livelli di logica, il demux un livello di logica, i due mux due livelli di logica ciascuno mentre il sommatore richiede 64 livelli di logica (32 full-adder ad un bit a cascata, ciascuno con due livelli di logica). Complessivamente il ciclo di clock richiede di stabilizzare almeno 66 livelli di logica a cascata (due del mux e 64 del full-adder con questa implementazione naif).

Esercizio 3

1) Dei 48 bit dell'indirizzo logico, gli 8 bit meno significativi ci danno l'indice di segmento, i seguenti 10 bit ci danno l'indice della tabella di primo livello. Rimangono 30 bit. I 12 bit più significativi sono l'offset all'interno della pagina mentre i restanti 18 bit sono divisi equamente in 9 bit per l'indice della tabella delle pagine di secondo livello e 9 bit per l'indice della tabella di terzo livello.



2) L'indirizzo fisico è dato dai 24 bit contenuti nella tabella delle pagine dell'ultimo livello (che riferiscono l'indirizzo base di una pagina fisica) più i bit dell'offset (12 bit). Quindi l'indirizzo fisico è formato da 36 bit. La massima memoria fisica indirizzabile è 2^{36} byte = 64 GB.

3) Il segmento S è formato da 314573 pagine logiche ($\lceil \frac{1.2*2^{30}}{2^{12}} \rceil$). Dato che si possono indicizzare $512 * 512 = 2^{18}$ pagine per ogni entry della tabella di primo livello, ci serviranno 2 entry della tabella di primo livello per indirizzare tutte le pagine del segmento (quelle di indice 0 ed di indice 1). Il numero di tabelle di secondo livello utilizzate è pari a 2 dato che $2^{2*9} < 314573 < 2^{3*9}$.
Nello specifico, la tabella di secondo livello di indice 0 sarà tutta utilizzata (i 512 puntatori alle tabelle di terzo livello sono tutti validi); la tabella di secondo livello di indice 1 sarà utilizzata solo parzialmente, in particolare saranno valide solo le prime 103 entry ($\lceil \frac{314573-(512*512)}{512} \rceil$), ossia quelle di indice nel range [0;102]. La tabella di terzo livello puntata dalla entry con indice 102 della tabella di secondo livello, ha valide solo le prime 205 entry (314573 – [(512*512) + 102*512]), ossia quelle di indice nel range [0;204].
4) L'indirizzo logico 0x20004CCCD100 appartiene sicuramente al segmento S. L'indirizzo genera una eccezione di violazione di segmento perché indirizza una pagina non appartenente al segmento al di fuori del limite superiore del segmento (cioè al di fuori del bound che corrisponde all'indirizzo esadecimale 0x20004CCCCFFF).
Infatti, trasformando l'indirizzo in binario e scomponendo l'indirizzo in base al formato degli indirizzi logici del sistema, abbiamo:
0x20004CCCD100 = (00100000 0000000001 001100110 011001101 000100000000)
- l'indice di segmento è valido 32<256;
- l'indice idx1 corrisponde all'entry 1 della tabella di primo livello, il cui puntatore è definito;
- l'indice idx2 corrisponde all'entry 102 della tabella di secondo livello (ultima entry valida);
- l'indice idx3 corrisponde al valore 205 che è una entry non valida (la tabella di terzo livello per tale entry non è caricata in memoria) perché gli indici di entry validi sono solo quelli nel range [0;204].

Esercizio 4

Si utilizzano i seguenti semafori: *mux* inizializzato ad 1, *waitT1T2* e *waitT3T4* inizializzati a 0. Inoltra si usano i seguenti contatori tutti inizializzati a 0:

- *T1T2* numero di scrittori in sezione critica (contatore binario);
- *T1T2waiting* numero di scrittori in attesa;
- *T3T4* numero di lettori in sezione critica;
- *T3T4waiting* numero di lettori in attesa.

```
accediT1T2() {
  bool ok=false;
  P(&mux);
  if (T3T4==0 && T1T2==0) {
    T1T2=1;
    ok=true;
  } else T1T2waiting++;
  V(&mux);
  if (!ok) P(&waitT1T2);
}
```

```
accediT3T4(&msg) {
  bool ok=false;
  P(&mux);
  if (T1T2waiting==0 && T1T2==0) {
    T3T4++;
    ok=true;
  } else T3T4waiting++;
  V(&mux);
  if (!ok) P(&waitT3T4);
}
```

```
rilasciaT1T2() {
  P(&mux);
  T1T2=0;
  If (T1T2waiting>0) {
    T1T2=1;
    T1T2waiting--;
    V(&waitT1T2);
  } else {
    while(T3T4waiting>0) {
       T3T4waiting--;
       V(&waitT3T4);
    }
  }
  V(&mux);
}
```

```
rilasciaT3T4() {
   P(&mux);
   T3T4--;
    if (T3T4==0 && T1T2waiting > 0) {
       T1T2=1 ;
       T1T2waiting-- ;
       V(&waitT1T2) ;
    }
   V(&mux);
 }
```