Esame di **Architetture degli Elaboratori** - 26/01/2021

A.A. 2020-21

Cognome e Nome _____ Matricola _____

Soluzione (ogni domanda vale **3 punti**, la 14 vale **9 punti**)

Per tutta la verifica, **N** sarà uguale alle cinque o sei cifre del numero della matricola dello studente, dapprima privata di eventuali lettere, e poi trascritta nel verso opposto aggiungendo infine zeri fino a raggiungere un numero di sei cifre.
Es.: se la matricola è 237424, allora **N = 424732**
se la matricola è 237400, allora **N = 473200**
se la matricola è I-37424, allora **N = 424730.**

1. Si converta N in base 4
Il numero va diviso per 4 fino a quando, giunti alla k-esima divisione, il risultato $N/4^k$ non diventa minore di 4. La conversione è uguale alla successione $R_kR_{k-1}...R_1$ dei k resti $R_1$, $R_2$, ..., $R_k$ delle divisioni partendo dall'ultima fino alla prima eseguita

| | |
|---|---|
| N | |
| N/4 | $R_1$ |
| N/16 | $R_2$ |
| . | . |
| . | . |
| $N/4^k$ | $R_k$ |

Risultato: $\mathbf{R_kR_{k-1}...R_1}$

2.
a) Si prendano le tre cifre **meno** significative di N e le si converta in binario. Risultato: $\mathbf{b_9b_8...b_0}$
b) Si prendano le tre cifre **più** significative di N e le si converta in binario. Risultato: $\mathbf{c_9c_8...c_0}$
c) Si sottragga **in complemento a due a 11 bit** il risultato ottenuto al punto b) al risultato ottenuto al punto a)

Calcolato il complemento a 2 del numero binario $c_9c_8...c_0$, che non può essere più esteso di 10 bit, immediatamente il risultato è dato da

$$\begin{array}{r} b_9b_8...b_0 + \\ 1d_9d_8...d_0 = \\ \hline e_{10}e_9e_8...e_0 \end{array}$$

in cui $1d_9d_8...d_0$ è il complemento a 2 (sempre negativo) del numero binario $c_9c_8...c_0$. Si noti che essendo la somma svolta in aritmetica a 11 bit tra due numeri di segno opposto, il calcolo non può in ogni caso dar luogo a overflow.

Risultato: $\mathbf{e_{10}e_9e_8...e_0}$

3. Un chip di memoria ha una capacità di N kB. Se ogni anno la capacità di memoria triplica a parità di area del chip, e la capacità del chip viene aggiornata ogni sei mesi, quanti semestri sono trascorsi da quando un chip avente la stessa area ha oltrepassato la capacità di 64 kB ?

Sei mesi prima la memoria aveva una capacità minore di un fattore $\sqrt{3} = 3^{1/2}$. Dunque, occorre trovare **il più piccolo numero intero** k di semestri che soddisfano la disequazione $N/3^{k/2} < 64$

Risposta: **k intero tale che $N/3^{k/2} < 64$**

4.
a) Si calcoli il **resto intero di N/256** e lo si converta in un numero binario a 8 bit. Risultato: $\mathbf{b_7...b_0}$
b) Si costruisca una tabella di verità a 3 ingressi, la cui unica uscita assume valori uguali alle cifre costituenti il risultato al punto a) a partire dalla cifra meno significativa a quella più significativa nel verso crescente del valore degli ingressi.

Risposta:

| A | B | C | E |
|---|---|---|---|
| 0 | 0 | 0 | $b_0$ |
| 0 | 0 | 1 | $b_1$ |
| 0 | 1 | 0 | $b_2$ |
| 0 | 1 | 1 | $b_3$ |
| 1 | 0 | 0 | $b_4$ |
| 1 | 0 | 1 | $b_5$ |
| 1 | 1 | 0 | $b_6$ |
| 1 | 1 | 1 | $b_7$ |

5. Si supponga di realizzare la tabella di verità al punto 4 come una rete combinatoria in **forma canonica dualizzata**, cioè formata dall'AND di porte logiche OR, entrambe non necessariamente binarie.
a) Di quante porte OR è composta la rete? Risposta: _____
b) Di quante porte AND è composta la rete? Risposta: _____
c) Di quanti transistor è composta la rete? Risposta: _____

Per ciascun bit $b_i$ uguale a 0 si deve prendere una porta OR a 3 ingressi alimentata dai segnali d'ingresso che definiscono la riga i-esima, negando quelli uguali a 1. Fatto ciò, l'uscita da

ciascuna delle k porte OR alimenta un ingresso di una porta AND a k ingressi. A questo punto è sufficiente ricordare che
- ogni porta NOT contiene 1 tranistor
- ogni porta OR a 3 ingressi contiene 4 transistor
- la porta AND a k ingressi contiene k+1 transistor.

6. Si definisca una procedura **algoritmica** realizzabile da un semplice circuito sequenziale che, avendo accesso a ciascun bit di una codifica binaria di parità dispari a M bit, in cui M è la cifra più significativa in N, permette la verifica da parte del ricevitore della correttezza di ogni sequenza di M bit ricevuta.

Risposta: (per esempio) **eseguo la somma a un bit di ciascuno degli M bit che formano la sequenza, scartando il riporto; se il risultato è 1 accetto la codifica**. Oppure: **eseguo l'OR esclusivo $b_{M-1}$ XOR $b_{M-2}$ XOR ... XOR $B_1$ XOR $b_0$ degli M bit che formano la sequenza; se il risultato è 1 accetto la codifica.**

7. Un bus parallelo a 4 linee ha una banda passante di N bit/s. Se il segnale digitale su ogni linea può assumere 2^M ("2 alla M") livelli, in cui M è la cifra più significativa in N, qual è la frequenza minima ammessa per il bus considerato?

Risultato: **è il numero intero più piccolo maggiore di N/4/M.**

8. Su una scheda Arduino UNO è stato montato un clock difettoso che oscilla con frequenza pari a N Hz in luogo dei previsti 16 MHz. Se si utilizza il Timer0 (8bit) con fattore di prescaling uguale a 8, ogni quanti millisecondi (ms) andrà in overflow il timer in questione ?

Avremo un tick del clock ogni 1/N s. Quindi con il Timer0 a 8 bit, che conta da 0 a 255, e il prescaler posto a 8 avremo overflow dopo 1/N * 8*256 s.

Risultato: **1/N * 8*256*1000 ms**

9. Una memoria cache ad accesso immediato è formata da 2048 entry numerate rispettivamente da 0 a 2047, ciascuna di 32 Byte numerati rispettivamente da 0 a 31. Quale entry e quale byte occuperà il contenuto della locazione di memoria di indirizzo N (compreso tra 0 a 999999) quando è presente in cache ?

La locazione in questione occupa lo slot R nella cache ottenuto dal resto R della divisione intera di N/(32*2048): R= N/(32*2048). L'entry si trova dal **risultato della divisione intera di R/32.** Infine, il byte si trova dal **resto della divisione intera di R/32.**

10. Un'ISA (Instruction Set Architecture) a 32 bit contiene un'istruzione JUMP che permette un salto relativo di al più N word, sia in avanti che all'indietro. Quanti bit restano al massimo liberi per il codice operativo dell'istruzione in questione ?

Per saltare in avanti o all'indietro di N word occorre specificare un valore, la cui rappresentazione binaria occupa un numero di bit uguale all'**intero più piccolo superiore a $\log_2 N + 1$**, in cui il bit addizionale serve per memorizzare il segno positivo o negativo. I restanti **32 - intero($\log_2 N + 1$) bit restano disponibili.**

11 [INF]. Dette rispettivamente Nsx la cifra **più** significativa di N e Ndx la cifra **meno** significativa di N, convertire il numero 2^Nsx + 2^-Ndx ("2 alla Nsx più 2 alla meno Ndx") in codifica *floating point IEEE 754* a 32 bit.

Immediatamente risulta un numero binario nella forma 100...0.0.....001, contenente una parte intera formata da uno e Nsx zeri, e una parte decimale formata da Ndx-1 zeri e uno. A questo punto il numero è scritto in forma esponenziale: 1.0...001E(Nsx), da cui immediatamente la codifica richiesta convertendo il valore Nsx in notazione binaria eccesso 127.

12 [INF]. Detti rispettivamente A, B e C gli ingressi, si completi il disegno qui sotto in modo da realizzare la mappa di Karnaugh che minimizza l'espressione booleana relativa alla tabella di verità già calcolata all'esercizio 5.

Detti b0, ..., b7 le uscite calcolate all'esercizio 5 si ha

| **AB** | **00** | **01** | **11** | **10** |
|---|---|---|---|---|
| **C** | | | | |
| **0** | **b0** | **b1** | **b3** | **b2** |
| **1** | **b4** | **b5** | **b7** | **b6** |

13 [INF]. Di quanti **nodi** e di quanti **archi** si compone una macchina di Mealy definita sull'insieme delle cifre decimali I={0,1,...,9}, in grado di riconoscere **solo la prima occorrenza** del numero N da una sequenza di cifre decimali indefinitamente lunga?

Ogni nodo ha 10 archi in uscita. Di questi, solo uno etichetta la cifra corretta appartenente a N. Se, dunque, N possiede 6 cifre, cioè $N = N_5N_4N_3N_2N_1N_0$, la macchina **deve** andare dal nodo $q_i$ al nodo $q_{i+1}$ non appena le prime i cifre lette sono corrette e la cifra $N_i$ letta dalla sequenza è a sua volta corretta. Solo l'arco che connette il nodo $q_5$ al nodo $q_6$ è etichettato con un valore che segnala l'avvenuto riconoscimento del numero (es.: 1 se tutti gli altri

sono etichettati con 0). Il riconoscimento dunque necessita di 6 nodi, escluso il nodo iniziale in cui la macchina resta fino a quando l'ingresso è diverso da $N_0$. Poichè solo la prima occorrenza è riconosciuta, tutti gli archi del nodo finale $q_6$ formano dei cappi chiusi sullo stesso nodo e, quindi, la macchina non cambia più stato. Tuttavia, si noti che era inutile ragionare su quale nodo finisse ogni arco perchè il numero di archi non può cambiare. In definitiva dunque abbiamo

Risultato: **1+6 = 7 nodi e quindi 7*10 = 70 archi.**

14 [INF]. Scrivere un programma in assembly per ARM il quale, letto il valore N da un word in memoria principale, è in grado di determinare se ciascuna cifra in N è minore o uguale a 5 oppure maggiore di 5. Il programma restituirà in ciascun elemento di un array il valore -1 oppure 1 se la corrispondente cifra in N è rispettivamente minore o uguale a 5 oppure maggiore di 5. E' utile appoggiarsi alla subroutine riportata qui sotto, la quale esegue la divisione intera R0/R1 tra gli interi positivi contenuti in R0 e R1, restituendo in R0 il resto della divisione intera e in R1 il risultato della stessa:

```
divisione:                          ; r0 / r1 (interi positivi)
                                    ; resto in r0, risultato in r1
          mov r3, #31               ; contatore
          mov r2, #0                ; inizializza r2 a 0
loop:     mov r2, r2, lsl #1        ; risultato parziale
          cmp r1, r0, lsr r3        ; se r1 < (r0>>) ...
          suble r0, r0, r1, lsl r3  ; ...sottrai <<r1 a r0
          addle r2, r2, #1          ; aggiorna risultato parziale
          subs r3, r3, #1           ; aggiorna contatore
          bge loop
          mov r1, r2
          mov pc, lr
```

É gradita la presenza di commenti al codice prodotto.

```
@ *************** data segment ***************
.data

n:          .word 743563
divisore:   .word 10
soluzione:  .skip 4*6

@ *************** code segment ***************
.text

main:
        ldr r0, =n
```

```
        ldr r0, [r0]              ; load input number
        ldr r4, =divisore         ; r4 points to divider
        ldr r5, =soluzione        ; r5 points to solution
        mov r6, #1                ; load output
        mov r7, #-1               ; load output
loop_main:
        ldr r1, [r4]              ; load divider
        bl divisione              ; compute remainder and result
        cmp r0, #5                ; if remainder > 5...
        strgt r6, [r5], #4        ; ...store #1
        strle r7, [r5], #4        ; ... else store #-1
        movs r0, r1               ; if result != 0...
        bne loop_main             ; ...then loop again
exit:
        swi 0x11

@@@@@@@@@@ subroutine divisione @@@@@@@@@@@@@@@

divisione:                        ; r0 / r1 (interi positivi)
                                  ; resto in r0, risultato in r1
         mov r3, #31              ; contatore
         mov r2, #0               ; inizializza r2 a 0
loop:
         mov r2, r2, lsl #1       ; risultato parziale
         cmp r1, r0, lsr r3       ; se r1 < (r0>>) ...
         suble r0, r0, r1, lsl r3 ; ...sottrai <<r1 a r0
         addle r2, r2, #1         ; aggiorna risultato parziale
         subs r3, r3, #1          ; aggiorna contatore
         bge loop
         mov r1, r2
         mov pc, lr
```