Università degli Studi di Udine

DMIF — Dipartimento di Scienze Matematiche, Informatiche e Fisiche

Esame **B** di **Architetture degli Elaboratori**

Soluzione

A.A. 2017-18 — II appello — 20 febbraio 2018

N.B.: il punteggio associato ad ogni domanda è solo una misura della difficoltà, e peso, di ogni domanda. Per calcolare il voto complessivo bisogna normalizzare a 30 (circa).

1. Convertire il valore $5.\overline{1}$ dalla base 10 alla base 9.

   R: (3 pt)

   ```
   5 | 9      .111.. | 9
   -------     ----------
   0 | 5           0 | 1
   ```

   e quindi $5.\overline{1} = 5.1_9$. Infatti, $9^{-1} = 1/9 = 0.\overline{1}$, da cui $5.\overline{1} = 5 \cdot 9^0 + 1 \cdot 9^{-1}$.

2. Sono date le seguenti codifiche in complemento a 2 a 8 bit: $n_1 =$ `10110111`, $n_2 =$ `11001100`. Si calcoli la differenza $n_2 - n_1$ e, se possibile, si esprima il risultato nella stessa codifica.

   R: (3 pt) Complementando n_1 per cambiarne il segno: $-n_1 =$ `01001001`, eseguiamo successivamente la somma

   ```
    11001100 +
    01001001 =
   ---------
   100010101
   ```

   che, coinvolgendo un valore positivo e uno negativo, dà falso *overflow* ed è quindi codificata come `00010101`, scartando appunto il nono bit.

3. [INF] Fornire il risultato dell'esercizio precedente in codifica *floating point* IEEE 754 a 32 bit.

   R: (3 pt) Il risultato trovato sopra può essere subito messo nella forma `1.0101E4`. La codifica richiesta avrà dunque bit di segno nullo, esponente uguale a $127 + 4 = 131 =$ `10000011`$_2$ e infine mantissa uguale a `0101`. Sistemando sui 32 bit previsti dallo standard IEEE 754 e convertendo alla base esadecimale:

   ```
   0|1 0 0 0 0 0 1 1|0 1 0 1 0 0 0 0 ...
      4   |   1   |   A   |   8   | 0 ...
   ```

   da cui la codifica richiesta: `0x41A80000`.

4. Si esprima il numero di bit presenti in una memoria a 64 GB come potenza di 2.

   R: (3 pt) Ricordando che 1 GB $= 2^{30}$ byte: $8 \cdot 64 \cdot 2^{30} = 2^3 \cdot 2^6 \cdot 2^{30} = 2^{39}$ bit.

5. Adoperando le regole di equivalenza booleana, calcolare quanto vale E nell'espressione seguente:

   $E = \overline{\overline{A} + \overline{B} + \overline{C}}\ \overline{A + \overline{C}}$

   R: (3 pt) Sfruttando le regole di De Morgan si ha $E = (ABC)(\overline{A}C)\ = ABC\overline{A}C\ = A\overline{A}BC = 0BC = 0$

6. [INF] Verificare il risultato ottenuto sopra con una mappa di Karnaugh.

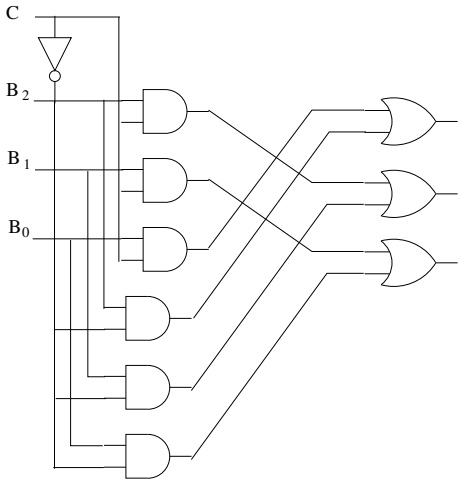
   R: (3 pt) La tabella di verità porge una mappa che presenta solo simboli `0`:

   ```
   BC    00  01  11  10
   A

   0      0   0   0   0
   1      0   0   0   0
   ```

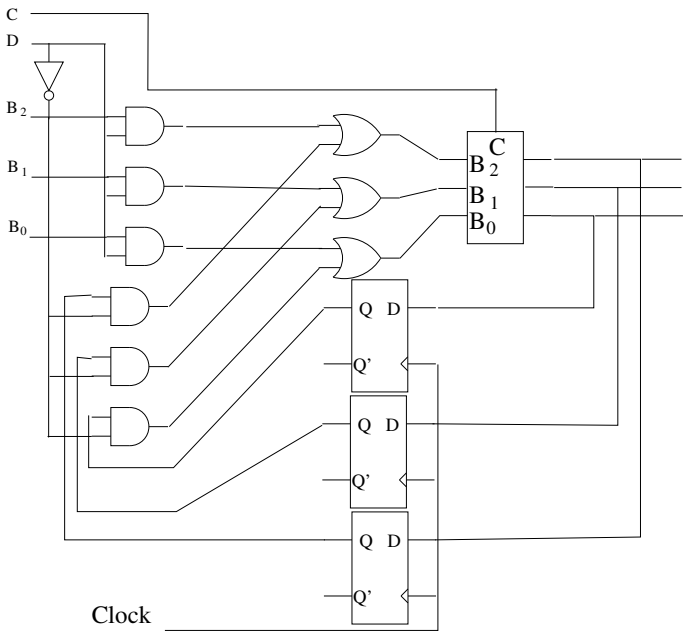
7. Progettare un traslatore rotazionale a 3 bit a destra, cioè una rete combinatoria in grado di ruotare un ingresso a 3 bit (B_2, B_1, B_0) verso destra di un bit oppure no, rispettivamente se un controllo C a un bit è asserito oppure no. In altre parole, se $C = 1$ allora (B_2, B_1, B_0) ruota in (B_0, B_2, B_1).

   R: (3 pt)



8. [INF] Avendo a disposizione il traslatore di cui sopra, che per comodità può essere denotato come un unico blocco, progettare un circuito sequenziale che, a seconda del valore assunto dal controllo C, a ogni ciclo di clock esegue la rotazione di un bit a destra oppure no dei bit (B_2, B_1, B_0) in ingresso oppure dei tre bit in uscita all'istante precedente. Quest'ultima decisione viene presa in base al valore assunto all'istante corrente da un bit etichettato come D.

   R: (3 pt)



9. Un codice a ripetizione tripla del carattere codifica le cifre decimali 0,1,. . . ,9 in corrispondenti triplette di cifre identiche. Il decodificatore restituisce la cifra c se almeno due delle tre cifre costituenti la tripletta sono uguali a c, altrimenti segnalando un errore di trasmissione. Si dica in quanti casi il codice riesce a correggere l'errore rispetto alla totalità delle codifiche che possono essere ricevute.

   R: (3 pt) Su 1000 possibili triplette ricevute, una è quella corretta. Ventisette triplette contengono due cifre identiche corrette. Quindi, in 27 casi su 999 una trasmissione errata viene corretta con successo.

10. Un elementare sistema di I/O programmato prevede che ogni 100 ms la CPU impegni 6 ms del proprio tempo nel *polling* di un'unica periferica. Se la periferica non deve essere servita questo tempo viene

sprecato, viceversa in caso di servizio i 6 ms vengono impiegati utilmente e la CPU lavora globalmente con un'efficienza del 100%. In queste ipotesi si calcoli quante volte la periferica dovrebbe mediamente richiedere un servizio affinchè la CPU lavori con un'efficienza del 97%.

R: (3 pt) Posto x il numero medio di volte in cui la periferica richiede di essere servita, se la periferica non richiede mai il servizio ($x = 0$) l'efficienza è del $100 - 6 = 94\%$. Al contrario, se richiede sempre un servizio ($x = 1$) l'efficienza sale al 100%. Il numero medio di volte si ottiene dunque imponendo che

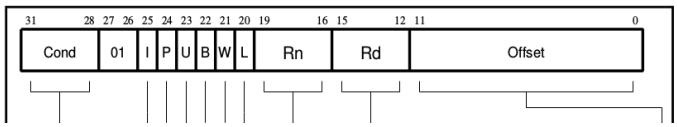
$94 \cdot (1 - x) + 100 \cdot x = 97 \Rightarrow 94 - 94x + 100x = 97 \Rightarrow 6x = 3 \Rightarrow x = 1/2$, cioè una volta su due.

11. Qual è la richiesta media di servizi al minuto che la periferica non può oltrepassare nel sistema di I/O presentato in precedenza?

    R: (3 pt) 600 servizi per ogni minuto.

12. I campi in figura appartengono a un'istruzione macchina per ARM che prevede la possibilità di salvare il contenuto di un registro `Rd` nell'indirizzo di memoria specificato dal contenuto del registro `Rn` corretto da un valore di *offset* presente nell'omonimo campo. Se *offset* vale 2^8, il contenuto del registro `Rn` può essere maggiore di *offset* in base alle informazioni presenti in figura?

    ARM Processor Instruction Set

    R: (3 pt) Sì, in quanto dalla stessa figura si evince che il registro `Rn` (per esempio il registro `Rn = 7`) è di 32 bit.

13. Quali locazioni di una memoria principale di 64 kB può contenere la linea 2 di una *cache* a 32 byte composta da 1024 *entry*?

    R: Le locazioni dalla $32 \cdot 2 = 64$ alla $32 \cdot 2 + 31 = 95$ e dalla $64 + 32 * 1024 = 32832$ alla $95 + 32 * 1024 = 32863$.

14. [INF] Scrivere un programma in assembly per ARM il quale, caricati due numeri m e $n > 0$ rispettivamente nei registri `r2` e `r3`, calcola il valore $(m+n-1)!/(m-1)! = (m+n-1)(m+n-2)\ldots(m+1)m$ attraverso una procedura ricorsiva che esegue le $n-1$ moltiplicazioni e deposita il risultato nel registro `r1`.

    R: (9 pt)

```
        .text
main:
        mov r2, #5              ; read m
        mov r3, #3              ; read n
        subs r3, r3, #1         ; will multiply n-1 times
        blne fattoriale         ; if n!=0 then call
        mov r1, r2              ; copy result in r1
        swi 0x11                ; exit program

fattoriale:
        stmfd sp!, {r4, lr}     ; save registers
        mov r4, r2              ; m in r4
        add r2, r2, #1          ; m <- m+1
        subs r3, r3, #1         ; decrement r3
        blne fattoriale         ; if r3!=0 call again
        mul r2, r4, r2          ; otherwise multiply
        ldmfd sp!, {r4, lr}     ; restore registers
        mov pc, lr              ; restore PC

        .end
```