Università degli Studi di Udine

DMIF — Dipartimento di Scienze Matematiche, Informatiche e Fisiche

Esame **A** di **Architetture degli Elaboratori**

Soluzione

A.A. 2018-19 — I appello — 29 gennaio 2019

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 32.

1. Convertire il valore $3030.\overline{030}_4$ alla base 16.

   R: (3 pt) Trattandosi di basi che sono una potenza di due, conviene dapprima convertire alla base 2 e poi raggruppare le quartine risultanti per riscrivere il valore nella base 16, tenendo conto del periodo nella parte decimale:

   $3030.\overline{030}_4 = 3030.\overline{030030}_4 = 30|30.\overline{03|00|30}_4 = \underbrace{1100_2}_{C_{16}}|\underbrace{1100_2}_{C_{16}}.\overline{\underbrace{0011_2}_{3_{16}}\underbrace{0000_2}_{0_{16}}\underbrace{1100_2}_{C_{16}}} = CC.\overline{30C}_{16}$

2. Sono date le seguenti codifiche in complemento a 2 a 8 bit: $n_1 = \texttt{10001000}$, $n_2 = \texttt{00001000}$. Si calcolino risultato ed eventuale resto della divisione $n_1 : n_2$ e, se possibile, si esprima il risultato nella stessa codifica.

   R: (3 pt) Per sicurezza conviene complementare n_1 per cambiarne il segno negativo: $-n_1 = \texttt{01111000}$. Eseguiamo ora la divisione che, essendo il divisore una potenza di due, coincide con una translazione verso destra del dividendo di un numero di posizioni uguale al numero di zeri del divisore:

   ```
   01111000:00001000 = 00001111, resto 00000000
   ```

   che, restituendo il segno originale, è ancora codificabile in complemento a 2 e vale $-\texttt{00001111} = \texttt{11110001}$. Si noti che la stessa operazione poteva essere eseguita direttamente nel dominio codificato.

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 $-1.111_2\texttt{E3}$. La codifica richiesta avrà dunque bit di segno asserito, esponente uguale a $127+3 = 130 = 10000010_2$ e infine mantissa uguale a 111_2. Sistemando sui 32 bit previsti dallo standard IEEE 754 e convertendo alla base esadecimale:

   ```
   1|1 0 0 0 0 0 1 0|1 1 1 0 0 0 0 0 ...
     C   |   1   |   7   |   0   | 0 ...
   ```

   da cui la codifica richiesta: `0xC1700000`.

4. Qual è la potenza di quattro che offre la miglior approssimazione di un Megabyte (cioè 1 MB = 10^6 byte)?

   Facoltativo e solo dopo avere terminato gli altri esercizi: Si dimostri l'affermazione appena fatta.

   R: (3 pt) La potenza in questione è 10. Infatti, la potenza di quattro che offre la migliore approssimazione di 1 kB è $2^{10} = 4^5$ byte, che prevede un errore di circa il 2%: $1000 \approx 2^{10}(1 - 0,02)$. La stessa considerazione vale per il MB in quanto quadrato del kB: $1000^2 \approx 2^{20}(1 - 0,02)^2 = 2^{20}(1 + 0,0004 - 0,04) \approx 2^{20}(1 - 0,04) = 4^{10}(1 - 0,04)$, quindi con un'approssimazione per difetto di circa il 4%.

5. Adoperando le regole di equivalenza booleana, ridurre l'espressione seguente in modo che ammetta una realizzazione adoperando esclusivamente porte NOT e porte OR a due ingressi:

   $E = \overline{ABC} + \overline{B\overline{D}}D$.

   R: (3 pt) $E = \overline{A}+\overline{B}+\overline{C}+(\overline{B}+\overline{\overline{D}})D = \overline{A}+\overline{B}+\overline{C}+\overline{B}D+D = \overline{A}+\overline{B}(1+D)+\overline{C}+D = (\overline{A}+\overline{B})+(\overline{C}+D)$.

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

   R: (3 pt) La tabella di verità permette quattro coperture di 4×2 simboli `1`, confermando il risultato dell'esercizio precedente:

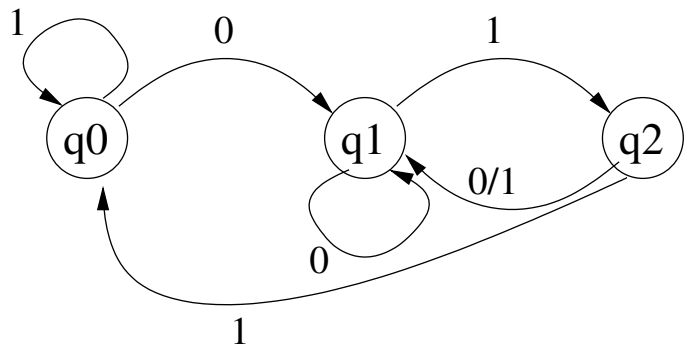
```
   AB    00  01  11  10
CD

00       1   1   1   1  |
01       1   1   1   1  | |
11       1   1   1   1    |
10       1   1   0   1
        -------
       <---         --->
```

7. Realizzare la rete booleana appena progettata adoperando un *multiplexer*, di cui non occorre esplicitare il circuito logico.

   R: (3 pt) É sufficiente scegliere un multiplexer a 16 ingressi e 4 controlli collegati rispettivamente ad `ABCD`, e poi connettere a una sorgente in tensione (per esempio 5V) tutti gli ingressi fuorchè quello associato al controllo `1110`, che invece sarà connesso alla massa 0V.

8. [INF] Progettare la macchina di Mealy che riconosce ogni sottostringa `010` contenuta in una sequenza definita sull'alfabeto $\mathcal{A} = \{0, 1\}$, in cui le sottostringhe non devono essere necessariamente disgiunte. La macchina restituisce il simbolo `1` solo quando riconosce la sottostringa, altrimenti restituendo `0` a fronte di ogni nuovo ingresso appartenente alla sequenza.

   R: (3 pt)



9. Si realizzi il codice binario più compatto, eventualmente a lunghezza variabile, in grado di codificare i 7 giorni della settimana etichettati come `lun,mar,mer,gio,ven,sab,dom`.

   R: (3 pt) Per esempio:

```
lun: 00
mar: 010
mer: 011
gio: 100
ven: 101
sab: 110
dom: 111
```

10. Un bus parallelo che trasmette un byte a ogni istante di clock è costituito da 8 linee seriali, ciascuna delle quali garantisce uno *skew* minore di 3 ns rispetto alla linea adiacente. Qual è la frequenza che il bus non può superare?

    R: (3 pt) Nel caso peggiore ogni bit può essere in ritardo di 3 ns rispetto a quello trasmesso sulla linea adiacente. Il bus deve quindi garantire un clock non inferiore a $T = 3 \cdot 7 = 21$ ns, più un'adeguata spaziatura tra l'ultimo bit e il primo bit del byte successivo, per comodità ancora di 3 ns. In totale risulta una frequenza $f = 1/T = 1/24$ GHz, equivalente a circa $41,66$ MHz.

11. Si spieghi che cos'è una porta del microcontrollore AVR di Arduino.

    R: (3 pt) É il canale di comunicazione logico tra il microcontrollore e il mondo esterno. A ogni porta corrisponde un set dei registri di memoria che ne rappresentano lo stato e le relative informazioni di input/output. A ogni pin (piedino fisico) del microcontrollore corrisponde un bit a una determinata posizione, per ogni registro relativo alla porta.

12. É possibile realizzare un ISA in cui le operazioni aritmetiche siano prive di argomenti?

    R: (3 pt) Sì. É sufficiente adoperare lo *stack* della memoria per eseguire qualunque operazione aritmetica, come fa per esempio il processore Mic1.

13. Sapendo che in una memoria paginata (dimensione locazione = 1 byte) ogni riga della *page table* è lunga 21 bit e il campo *offset* dell'indirizzo è di 7 bit, si dica quante pagine riescono a trovare spazio in memoria principale rispettivamente nei casi in cui il riempimento è ottimale (ovvero non c'è alcuna locazione libera) e pessimo (ovvero c'è il numero massimo di locazioni libere). Nel secondo caso il risultato può essere dato anche in forma approssimata.

    R: Dai dati si evince che la *page table* contiene indirizzi di 20 bit: di qui la dimensione della memoria principale uguale a $2^{20} = 1$ MB. Parallelamente la dimensione della pagina è data dal campo *offset* dell'indirizzo: $2^7 = 128$ byte. Se ne conclude che in condizioni di riempimento ottimale la memoria principale contiene $2^{20}/2^7 = 2^{13}$ pagine. Viceversa, quando il riempimento è il peggiore possibile allora ogni pagina è preceduta da 127 byte non occupati. In pratica, dunque, è come se ogni pagina occupasse $127 + 128 = 2^7 - 1 + 2^7 = 2^8 - 1$ locazioni. Di qui il calcolo del numero di pagine, uguale a

    $$\frac{2^{20}}{2^8 - 1} = 2^{12}\frac{2^8}{2^8 - 1} = 2^{12}\frac{256}{255} = 4096 \cdot \left(1 + \frac{1}{255}\right) = 2^{12} + 16.06,$$

    cioè $2^{12} + 2^4$.

14. [INF] Scrivere un programma in assembly per ARM il quale, caricati nei registri `r1` e `r2` rispettivamente due numeri n_1 e n_2 dalla memoria, dapprima verifica che ciascun numero occupi solo la metà meno significativa del registro che lo contiene. Se è così allora esegue l'algoritmo di moltiplicazione in colonna di due numeri positivi, infine depositando il risultato $n_1 \cdot n_2$ nel registro `r3`.

    R: (9 pt) Per $0 \leq i \leq 15$ il contenuto di `r1` traslato di i bit a sinistra è accumulato in `r3` se l'i-esimo bit di `r2` è uguale a uno.

```
        .data
n1:     .word 8
n2:     .word 6
        .text
main:
        ldr r1, =n1                ; read n1
        ldr r1, [r1]
        ldr r2, =n2                ; read n2
        ldr r2, [r2]
        tst r1, #0xFF00            ; test magnitude of n1
        bne end
        tst r2, #0xFF00            ; test magnitude of n2
        bne end
        mov r3, #0                 ; reset r3
        mov r4, #1                 ; bitwise mask
        mov r5, #0                 ; iterations counter
loop:
        tst r2, r4, lsl r5         ; r2 AND (r4 << r5)
        beq update                 ; if i-th bit is null skip sum..
        add r3, r3, r1             ; ..else add r1
update: mov r1, r1, lsl #1         ; shifts r1 leftward by one bit
        add r5, r5, #1             ; increment counter
        cmp r5, #16                ; if counter == 16..
        beq end                    ; ..exit loop..
        b loop                     ; ..else repeat
```

```
end:
        swi 0x11                    ; exit program
        .end
```