Università degli Studi di Udine

DMIF — Dipartimento di Scienze Matematiche, Informatiche e Fisiche

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

Soluzione

A.A. 2018-19 — IV appello — 16 luglio 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. Si converta il valore -2^{-18} nella base 16.

   R: (3 pt) É immediato convertire il valore in questione alla base 2:

   $-2^{-18} = -0.000000000000000001_2 = -1.0_2\mathtt{E}(-18)$.

   A questo punto, completando la quartina più a destra e raggruppando si ottiene immediatamente

   $-0.000000000000000100_2 = -0.00004_{16}$.

2. Ricordando la regola per il cambio di segno dei numeri codificati in complemento a due a 8 bit, si cambi il segno del numero zero verificando che la regola funziona anche in questo caso particolare. Successivamente si provi a cambiare il segno del più grande (in valore assoluto) numero negativo rappresentabile, e si motivi il risultato.

   R: (3 pt) La regola per il cambio di segno prevede di incrementare di uno la codifica complementare. Applicata allo zero si ottiene subito

   $-\mathtt{00000000} \Longrightarrow \mathtt{11111111} + \mathtt{1} = \mathtt{100000000}$, in cui si può eliminare il bit di riporto in quanto l'opposto dello zero è rappresentabile in complemento a due ed è lo zero stesso, ottenendo il risultato atteso.

   Applicando la stessa regola al più grande (in valore assoluto) numero negativo rappresentabile, che é -2^7, si ha

   $-\mathtt{10000000} \Longrightarrow \mathtt{01111111} + \mathtt{1} = \mathtt{10000000}$, che erroneamente restituisce ancora la codifica del numero -2^7. Infatti, il suo opposto 2^7 non é rappresentabile in complemento a due a 8 bit.

3. [INF] Convertire il numero 0.0625 in codifica *floating point* IEEE 754 a 32 bit.

   R: (3 pt) Il numero coincide con la frazione $1/16 = 2^{-4}$, e quindi può essere subito messo nella forma $1.0_2\mathtt{E}(-4)$. La codifica richiesta avrà dunque bit di segno non asserito, esponente uguale a $127 - 4 = 123 = 01111011_2$ e infine mantissa uguale a 0_2. Sistemando sui 32 bit previsti dallo standard IEEE 754 e convertendo alla base esadecimale:

   ```
   0|0 1 1 1 1 0 1 1|0 0 0 0 0 0 0 0 ...
      3   |   D   |   8   |   0   | 0 ...
   ```

   da cui la codifica richiesta: `0x3D800000`.

4. Si supponga che ogni cinque anni l'area di un chip di memoria diminuisca esponenzialmente, a parità di capacità, di un fattore 10. Un chip di 3 cm^2 che area doveva avere se prodotto 30 mesi prima, per contenere la medesima quantità di memoria?

   R: (3 pt) Poichè 30 mesi equivalgono a 2.5 anni, a parità di capacità l'area del chip doveva essere maggiore di un fattore $\sqrt{10} \approx 3.16$, ovvero misurare $3 \cdot 3.16 = 9.5$ cm^2.

5. Realizzare una porta NAND a tre ingressi adoperando porte NAND a due ingressi.

   R: (3 pt) É immediato scrivere $\overline{ABC} = \overline{\overline{AB}\,C}$. É altresì immediato negare una singola variabile attraverso una porta NAND: $\overline{X} = \overline{XX}$. Quindi, in definitiva,

   $\overline{ABC} = \overline{\overline{AB}\,C} = \overline{\overline{AB}\,\overline{AB}\,C}$, in cui $\overline{AB}$ di fatto é calcolato una sola volta.

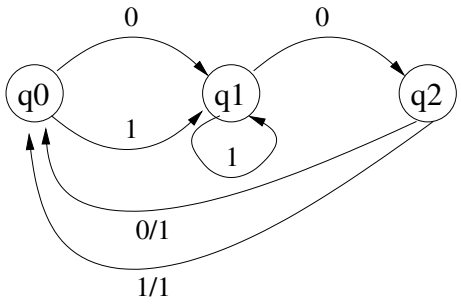
6. [INF] Di quanti transistor necessita la realizzazione più economica dell'espressione all'esercizio precedente?

   R: (3 pt) Una porta NAND a tre ingressi può essere costruita generalizzando lo schema della NAND a due ingressi, il quale necessita di due transistor. Quindi, la porta in questione può essere realizzata adoperando tre transistor posti in serie, cioè collegando l'emettitore del precedente al collettore del successivo.

7. Si scriva la tabella di verità del sommatore binario completo. Può questa tabella di verità essere utile per realizzare la porta NAND a tre ingressi vista all'esercizio 5? Se sì, come si può realizzare una porta NAND a tre ingressi adoperando il sommatore completo come parte fondamentale della realizzazione?

   R: (3 pt) É sufficiente adoperare i tre ingressi A, B e *Carry in* del sommatore completo, e collegare le sue uscite *Sum* e *Carry out* agli ingressi di una porta NAND per ottenere all'uscita di quest'ultima l'equivalente risultato di una porta NAND a tre ingressi.

8. [INF] Adoperando le porte logiche desiderate, realizzare la rete logica della macchina di Mealy in figura definita sull'alfabeto $\mathcal{A} = \{0, 1\}$. Nella stessa figura, s'intende che a ogni transizione l'uscita è uguale a 0 se non altrimenti specificato.



   R: (3 pt) Detti S_1 e S_0 due simboli appartenenti all'alfabeto necessari per specificare lo stato, I l'ingresso, E l'uscita, la tabella di verità porge

S_1	S_0	I	S_1^*	S_0^*	E
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	0	0	1
1	0	1	0	0	1
1	1	0	–	–	–
1	1	1	–	–	–

   da cui, aiutandosi eventualmente costruendo tre tabelle di Karnaugh una per ciascuna uscita,

   $S_1^* = \overline{S_1} S_0 \overline{I}$, $S_0^* = \overline{S_1}\,\overline{S_0} + \overline{S_1} I$, $E = S_1 \overline{S_0}$.

   A questo punto è immediata la costruzione di tre semplici sottoreti booleane, due delle quali generano rispettivamente S_1^* e S_0^* e sono collegate ciascuna a un flip-flop di tipo D per la produzione all'istante successivo di S_1 e S_0. La terza sottorete genera l'uscita E dalla macchina.

9. Un trasmettitore di simboli binari realizza un semplice codice a correzione d'errore, il quale da ogni simbolo da trasmettere produce una parola composta da 10 simboli identici a quello che di volta in volta dev'essere trasmesso. Qual è la distanza di Hamming n del codice appena realizzato? Qual è la decisione che deve prendere un ricevitore progettato per correggere un eventuale errore di trasmissione, se imponiamo che venga rispettata la proprietà fondamentale di correzione dell'errore di un codice con distanza di Hamming n?

   R: (3 pt) Poichè ogni simbolo binario viene trasformato in una parola di dieci simboli identici, la distanza di Hamming di un siffatto codice è $n = 10$. Un ricevitore progettato per correggere l'errore in una parola di questo tipo può decidere di sostituire in essa i simboli presenti in numero minore con quelli diversi presenti in numero maggiore. In tal modo, potendo correggere fino a $(n-1)/2 = 4$ simboli, il ricevitore rispetta la proprietà fondamentale di correzione dell'errore di un codice con distanza di Hamming n.

10. Un bus parallelo che funziona a una frequenza $f = 28$ MHz ha una banda passante di 336 Mbyte/s ed è composto da 24 canali seriali identici. Quanti bit invia ciascun canale durante ogni ciclo di clock?

    R: (3 pt) Ogni canale evidentemente ha una banda passante B uguale a $336/24 = 14$ Mbyte/s (o MB/s), ovvero di $14 \cdot 8 = 112$ Mbit/s (o Mb/s). Lavorando a $f = 28$ MHz, detto T il ciclo di clock ciascuno di questi canali invia $112 \cdot T = 112/f = 112/28 = 4$ bit durante ogni ciclo.

11. Qual é la motivazione principale per adoperare una piattaforma come Arduino piuttosto che una tradizionale architettura per PC in un'applicazione per l'IoT?

    R: (3 pt) Il motivo principale è legato al costo molto più basso della piattaforma in questione rispetto a un'architettura tradizionale, che avrebbe prestazioni inutilmente sovrabbondanti per applicazioni IoT. Similmente Arduino riduce i consumi energetici al minimo, esegue in modo deterministico le istruzioni e comunica direttamente con sensori e attuatori in modalità sia digitale che analogica tramite i GPIO.

12. Una semplice architettura a 16 bit prevede la presenza di un'istruzione `add R3,R1,R2`, sempre a 16 bit, la quale scrive nel registro `R3` la somma del contenuto dei registri `R1` e `R2`. Se il codice operativo (*opcode*) occupa 4 bit, qual è il numero massimo di registri specificabili come argomenti nell'istruzione in questione? Si motivi la risposta.

    R: (3 pt) Occupato nei 16 bit lo spazio per l'opcode, restano disponibili 12 bit da ripartire sui tre argomenti da specificare. La soluzione più ovvia è di riservare 4 bit per ogni campo argomento. In tal modo ogni campo può indirizzare al più $2^4 = 16$ registri diversi.

13. Un'architettura a 32 bit realizza una memoria *cache* il cui indirizzo virtuale è così composto: campo `TAG` 15 bit, campo `LINE` 12 bit, campo `WORD` 3 bit, campo `BYTE` 2 bit. Si consideri ora un ciclo che viene iterato consecutivamente per un certo numero di volte, in cui a ogni iterazione viene eseguita una lettura dalla locazione di memoria di indirizzo `r`, e poi immediatamente una scrittura nella locazione di memoria di indirizzo `w` come illustrato dallo pseudo-codice seguente:

```
        .
        .
        .
loop:
      read #r      ; legge un dato dalla locazione di indirizzo r
      write #w     ; scrive il dato nella locazione di indirizzo w
      bext fuori   ; esce dal ciclo se si verifica una condizione esterna
      jump loop    ; ..altrimenti ripete il ciclo
        .
        .
fuori:
        .
        .
```

    Si diano rispettivamente a `r` e a `w` due indirizzi numerici che provocano il 100% di *cache miss* a partire dalla seconda iterazione.

    R: Poichè il campo `WORD` é di 3 bit e il campo `BYTE` è di 2 bit, ogni linea di cache è lunga 32 byte. Poichè il campo `LINE` è di 12 bit, la cache possiede 4096 linee. Dunque, per esempio la prima linea contiene i byte 0-31, oppure 131072-131103 eccetera, in quanto $32 \cdot 4096 = 131072$ e $32 \cdot 4096 + 31 = 131103$. A questo punto, è sufficiente che `r` sia nel'intervallo 0-31 e `w` sia nell'intervallo 131072-131103 perchè ogni operazione di lettura/scrittura avvenga, a parte la prima di esse, inevitabilmente in un intervallo di locazioni di memoria che in quel momento non è presente nella cache.

14. [INF] Scrivere un programma in assembly per ARM il quale ricerca all'interno di una stringa presente nella memoria (direttiva assembly `.asciiz` seguita dalla stringa racchiusa tra doppi apici) un carattere anch'esso contenuto in memoria (stessa direttiva). Se questo carattere è presente nella stringa il programma restituisce nel registro `R4` l'indirizzo della locazione di memoria in cui il carattere è contenuto per la prima volta nella stringa. Altrimenti, il programma restituisce nello stesso registro il valore `-1`. Nota bene: un carattere occupa solamente un byte di memoria. É gradita la presenza di commenti al codice prodotto.

R: (9 pt)

```
        .data
stringa:
        .asciiz "ricerca"
carattere:
        .asciiz "e"

        .text
main:
        ldr r0, =carattere      ; char location in r0
        ldrb r2, [r0]           ; character to search in r2
        ldr r1, =stringa        ; string location in r1
loop:
        cmp r0,r1               ; if string and char location are equal..
        moveqs r4, #-1          ; ..then write -1 in r4, set registers..
        blt exit                ; ..and exit program
        ldrb r3, [r1],#1        ; string character in r3, increment pointer
        cmp r3,r2               ; if string char == search char..
        beq store               ; ..then go to store..
        b loop                  ; ..else repeat
store:
        mov r4, r1              ; store string char location
exit:
        swi 0x11                ; exit

        .end
```