Esercizi relativi all'architettura MIC-1

1. Nel MIC-1 ci vuole $1\,ns$ per preparare MIR, $1\,ns$ per mettere il contenuto di un registro sul bus B, $3\,ns$ per far funzionare l'ALU e lo shifter, e $1\,ns$ perchè i risultati ritornino ai registri. La larghezza dell'impulso di clock è di $2\,ns$. Questa macchina può funzionare a $100\,MHz$? E a $150\,MHz$? Quale è la frequenza massima di clock a cui la macchina può funzionare correttamente ?

   Soluzione:

   Il data path occupa $6\,ns$ così ripartiti:

   - $1\,ns$ per preparare MIR;
   - $1\,ns$ per mettere il contenuto di un registro sul bus B;
   - $3\,ns$ per far funzionare l'ALU e lo shifter;
   - $1\,ns$ perchè i risultati ritornino ai registri.

   Ai $6\,ns$ del data path bisogna aggiungere i $2\,ns$ della larghezza dell'impulso di clock. Quindi, per far funzionare correttamente la macchina, il periodo del clock deve essere superiore a $8\,ns$.

   A $100\,MHz$, il periodo del clock è:

   $$T = \frac{1}{100 \cdot 10^6} = \frac{1}{10^8} = 10\ ns$$

   ed è quindi sufficiente a far funzionare correttamente la macchina.

   A $150\,MHz$, il periodo del clock è:

   $$T = \frac{1}{150 \cdot 10^6} = \frac{1}{1.5 \cdot 10^8} = 6.667\ ns$$

   e non è, quindi, sufficiente a far funzionare correttamente la macchina.

   Dato che un ciclo di clock sull'hardware della macchina impiega $(6+2) = 8\ ns$, la frequenza massima di clock sarà:

   $$F = \frac{1}{T} = \frac{1}{8 \cdot 10^{-9}} = 125\ MHz$$

2. Dare due diverse traduzioni IJVM per la seguente istruzione, e commentare la loro validità:

   $$i = j + k + 4$$

   Soluzione: I traduzione:

```
ILOAD   j                   / carica j sullo stack
ILOAD   k                   / carica k sullo stack
IADD                        / carica j+k sullo stack
BIPUSH  4                   / carica 4 sullo stack
IADD                        / carica j+k+4 sullo stack
ISTORE  i                   / memorizza il risultato in i
```

II traduzione:

```
ILOAD   j                   / carica j sullo stack
ILOAD   k                   / carica k sullo stack
BIPUSH  4                   / carica 4 sullo stack
IADD                        / carica k+4 sullo stack
IADD                        / carica j+k+4 sullo stack
ISTORE  i                   / memorizza il risultato in i
```

ATTENZIONE! - se j ha valore negativo e k positivo, l'operazione $j + k + 4$ potrebbe essere ammessa, ma il risultato intermedio $k + 4$ dare overflow. In questo caso la seconda soluzione darebbe errore.

3. Quanto tempo occorre (espresso in ns) perchè una machhina MIC-1 con clock a $200\,MHz$ esegua l'istruzione JAVA:

$$i = j + k$$

Soluzione: Traduco l'istruzione in linguaggio IJVM e conto quante micrositruzioni occorrono per la sua esecuzione:

```
ILOAD   j               / impiega 6 microistruzioni
ILOAD   k               / impiega 6 microistruzioni
IADD                    / impiega 4 microistruzioni
ISTORE  i               / impiega 7 microistruzioni

Totale                      23 microistruzioni
```

Con un clock a $200\,MHz$, ogni microistruzione impiega un tempo pari a un periodo del clock:

$$\text{durata 1 microistruzione} = \frac{1}{200 \cdot 10^6} = 5 \text{ ns}$$

Per cui il tempo totale per l'esecuzione della istruzione JAVA data è:

$$5 \cdot 23 = 115\ ns$$

4. Esprimere in linguaggio MAL cosa esegue la seguente μistruzione MIC-1 (codificata per semplicità in esadecimale):

$$0\,0\,0\,0\,C\,2\,1\,0\,3$$

Soluzione: La decodifica (in binario) dei vari campi della μistruzione produce:

```
NEXTADDR        0 0 0 0 0 0 0 0 0
JAM             0 0 0
SH              0 0
ALU             0 0 1 1 0 0
C               0 0 1 0 0 0 0 1 0
W/R             0 0 0
B               0 0 1 1
```

il cui significato in MAL è:

$$\text{TOS} = \text{MDR} = \text{MBRU and H ; GOTO 0x000}$$

5. Si voglia aggiungere all'insieme di istruzioni ISA-IJVM la nuova istruzione POPTWO che toglie due parole dalla cima stack. Scrivere il microinterprete MIC-1 per questa nuova istruzione.

   Soluzione:

```
poptwo1                 SP=SP-1
poptwo2                 MAR=SP=SP-1; rd
poptwo3
poptwo4                 TOS=MDR; goto Main1
```

6. Si voglia aggiungere all'insieme di istruzioni ISA-IJVM la nuova istruzione:

   $$BI2PU \ < const16 >$$

   che fa il push sullo stack di $< const16 >$ che è una costante con segno a 16 bit (2 byte).
   Scrivere il microinterprete MIC-1 per questa nuova istruzione.

   Soluzione:

```
bi2pu1                  PC=PC+1; fetch
bi2pu2                  H=MBR << 8
bi2pu3                  MDR=TOS=MBRU OR H
bi2pu4                  MAR=SP=SP+1; wr
bi2pu5                  PC=PC+1; fetch; goto Main1
```

7. Il risultato ottenuto con la nuova istruzione $BI2PU \ < const16 >$, potrebbe anche essere ottenuto applicando alla istruzione $BIPUSH$ il prefisso $WIDE$, con l'intesa che $WIDEBIPUSH < const16 >$ ha un operando con segno a 16 bit (concordemente a quanto succede per $WIDEILOAD$ e $WIDEISTORE$).
   Scrivere il microinterprete MIC-1 per questa nuova istruzione.

   Soluzione:

```
wide1                PC=PC+1; fetch; goto (MBR or x100)

widebipush1          PC=PC+1; fetch
widebipush2          H=MBR << 8
widebipush3          MDR=TOS=MBRU OR H
widebipush4          MAR=SP=SP+1; wr
widebipush5          PC=PC+1; fetch; goto Main1
```

Discutere la differenza fra le due implementazioni.

8. Si vuole aggiungere all'insieme ISA-IJVM la nuova istruzione:

$$BISH8PU \ < byte >$$

che shifta di 1 byte verso sinistra la parola affiorante sullo stack e aggiunge l'operando $< byte >$ nel byte meno significativo. Scrivere il microinterprete MIC-1 per questa nuova istruzione.

Soluzione:

```
bish8pu1   MAR=SP                 // MAR punta alla cima dello stack
bish8pu2   H=TOS << 8             // shifta di 1 byte parola affiorante
bish8pu3   TOS=MDR=MBRU OR H;wr   // carica operando e aggiorna TOS
bish8pu4   PC=PC+1;fetch          // incrementa PC e fetch
bish8pu5   goto Main1             // termina
```