Programmazione e Architetture degli Elaboratori - Soluzioni Foglio 2

Luca Manzoni, Michele Rispoli, Pietro Morichetti

Esercizio 01

Disegnare la tabella di verità associata al seguente circuito logico 1.

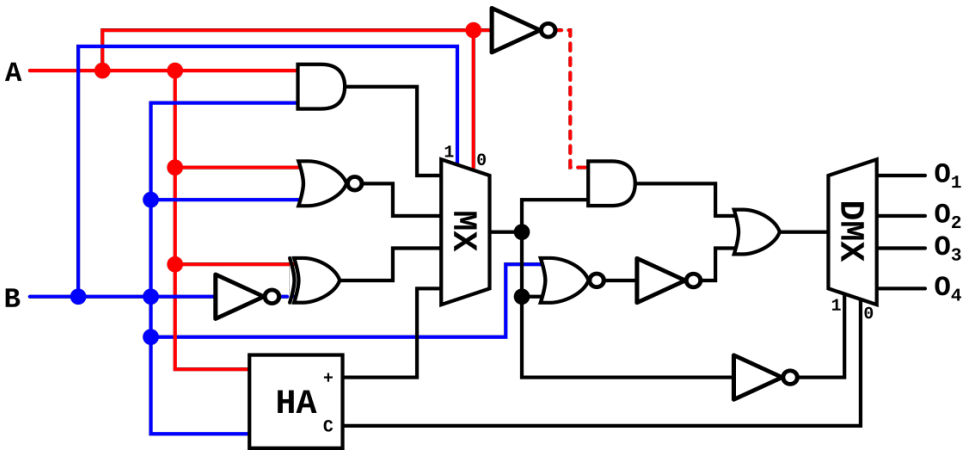

Figure 1: rete logica: esercizio 01

dove MX indica un multiplexer, DMX un demultiplexer, e HA un half-adder.

Soluzione:

A	B	O_1	O_2	O_3	O_4
0	0	0	0	0	0
1	0	0	0	0	0
0	1	0	0	1	0
1	1	0	0	0	1

Esercizio 02

Disegnare un circuito con 3 input (A, B, C) e due output (O_1, O_2) che realizza la seguente tabella di verità:

A	B	C	O_1	O_2
0	0	0	1	0
1	0	0	0	0
0	1	0	0	0
0	0	1	1	0
0	1	1	1	1
1	0	1	0	0
1	1	0	1	1
1	1	1	0	1

Soluzione: il circuito calcola le espressioni booleane

$$O_1 = A \iff (B \wedge \neg C)$$
$$O_2 = (A \vee C) \wedge B$$

che corrisponde al seguente circuito logico 2.

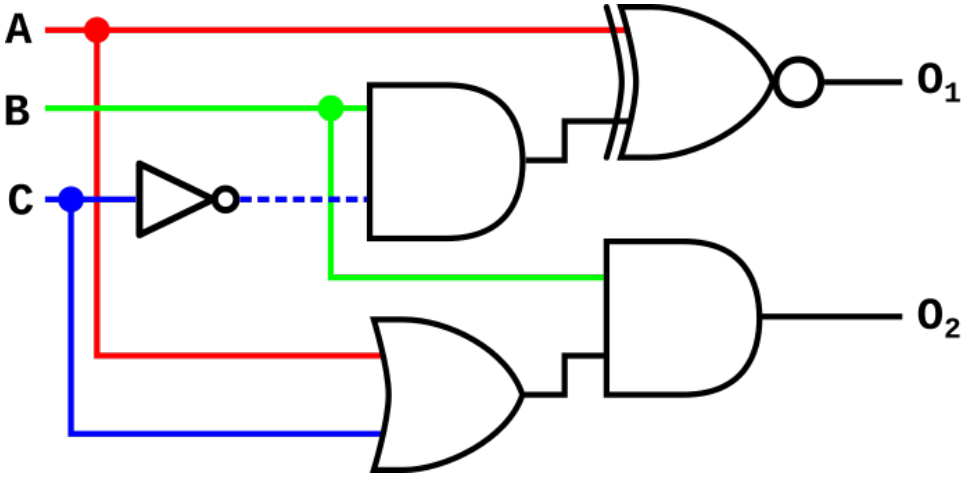

Figure 2: rete logica: soluzione esercizio 02

Esercizio 03

Considerati i registri `R0`, `R1` e `R2`, tradurre le seguenti descrizioni in istruzioni assembly (ARM 32):

1. Salvare in `R0` la costante 32, ed in `R1` la costante 12

   Sol:

```
MOV     R0, #32
MOV     R1, #12
```

2. Salvare in `R0` la somma tra `R0` e `R1`

   Sol:

```
        ADD     R0, R0, R1
```

3. Copiare in `R2` il contenuto di `R0` se il contenuto di `R0` è strettamente maggiore di `R1`

   Sol:

```
        CMP     R1, R0
        BLE     exit
        MOV     R2, R0
exit
```

4. Sottrarre a `R2` il valore in memoria all'indirizzo `0xA00` e, se il risultato è uguale al valore in `R1`, salvare il valore di `R0` in memoria all'indirizzo `0xB00`. (nota: potete utilizzare altri registri per l'indirizzo di memoria ed il valore che vi troviamo)

   Sol:

```
        MOV     R4, #0xA00
        LDR     R3, [R4]
        SUB     R2, R2, R3
        CMP     R1, R2
        BNE     exit
        MOV     R4, #0xB00
        STR     R0, [R4]
exit
```

5. Salvare in `R3` il risultato della moltiplicazione tra `R1` e `R2` (supponi che i valori di `R1` e `R2` strettamente positivi)

   Sol:

```
        MOV     R3, #0
loop    CMP     R1, #0
        BEQ     exit
        ADD     R3, R3, R2
        SUB     R1, R1, #1
        B       loop
exit
```

Esercizio 04

Tradurre le seguenti istruzioni assembly (ARM 32) in linguaggio "umano":

1.
```
MOV     R0, #0x27
MOV     R1, #0x02
```

   Sol: Assegnare i valori esadecimali `0x27` a `R0` e `x02` a `R1`

2.
```
MOV     R2, R1
MOV     R1, R0
MOV     R0, R2
```

   Sol: Scambiare i valori dei registri `R0` e `R1` usando `R2` come buffer

3.
```
MOV     R3, #0x070
LDR     R0, [R3]
MOV     R3, #0x700
LDR     R1, [R3]
SUB     R2, R0, R1
MOV     R3, #0x770
STR     R2, [R3]
```

   Sol: Calcola la sottrazione tra i valori in memoria agli indirizzi `0x070` e `0x700` e salva il risultato in memoria a `0x770`

4.
```
        CMP     R4, #1
        BNE     opt2
        MOV     R0, #0
        MOV     R1, #1
        MOV     R2, #2
        B       exit
opt2    MOV     R0, #0
        MOV     R1, #0
        MOV     R2, #0
exit
```

   Sol: Imposta i valori di `R0`, `R1` e `R2` a 0, 1 e 2 se `R4` è 1, altrimenti li imposta tutti a 0

5.
```
MOV     R4, #0xAB0
LDR     R2, [R4]
CMP     R2, #0
BLT     opt2
```

```
        ORR     R3, R0, R1
        B       save
opt2    EOR     R3, R0, R1
save    MOV     R4, #0xBA0
        STR     R3, [R4]
```

Sol: Se il valore in memoria a `0xAB0` è ≥ 0 calcola `R0` OR `R1`, altrimenti calcola lo XOR, ed in entrambi i casi salva il risultato in memoria a `0xBA0`

Esercizio 05

Si consideri il seguente frammento di codice C e lo si traduca in un possibile corrispettivo codice assembly (Arm32). Usare solo i registri `R0`, `R1` ed `R2`. Al termine dell'esecuzione il valore di q deve trovarsi in `R2` mentre il valore di r deve trovarsi in memoria all'indirizzo `0xB00`.

```
int main(int argc, char** argv){

    int a = 10;
    int b = 4;
    int q = 0;
    int r = 0;

    q = a / b;
    r = a % b;
    return 0;
}
```

Sol:

```
        MOV     R0, #10 ;; a
        MOV     R1, #4 ;; b

        MOV     R2, #0 ;; q

loop    CMP     R0, R1
        BLT     exit
        SUB     R0, R0, R1
        ADD     R2, R2, #1
        B       loop
exit    MOV     R1, #0xB00
        STR     R0, [R1]
```

Esercizio 06

Si consideri il seguente frammento di codice assembly (ARM 32) e lo si traduca nel corrispettivo codice C. Per semplicità i nomi della variabili da usare in C sono indicati come commenti.

```
        MOV     R0, #0 ;; i
        MOV     R1, #0 ;; sum
        MOV     R2, #0

loop    CMP     R0, #5
        BGE     exit

        MOV     R3, R0
inl     CMP     R3, #0
        BLE     next
        ADD     R1, R1, R0
        SUB     R3, R3, #1
        B       inl

next    ADD     R0, R0, #1
        B       loop

exit
```

Sol:

```
int main(int argc, char** argv){

    int sum = 0;

    for(int i = 0; i < 5; i++){
        sum += i * i;
    }

    return 0;
}
```