Esercizi arm Libro Professore pag 192

1) prototipo: word MAX(word X, word Y, word Z)
pre-condizione: nessuna
post-condizione: restituisce il massimo tra X, Y e Z

```
MOV R0, #25; //X
MOV R1, #21; //Y
MOV R2, #22; //Z

CMP R0, R1;
BLT else;  //R0<R1
CMP R0, R2;
BLT else2;  //R0<R2
MOV R3, R0;  //R3=MAX(R0,R1,R2)=R0
B fine;
else:
            CMP R1, R2;
            BLT else3; //R1<R2
            MOV R3, R1; //R3=MAX(R0,R1,R2)=R1
            B fine;
else2:
            MOV R3, R2; //R3=MAX(R0,R1,R2)=R2
            B fine;
else3:
            MOV R3, R2; //R3=MAX(R0,R1,R2)=R2
            B fine;
fine:
            NOP;
```

2) prototipo: word MAX(word[] V)
pre-condizione: V[0]>0 (dimensione array>0)
post-condizione: restituisce l'elemento massimo dell'array

```
V: .word 5,3,10,8,9,1; //array, primo elemento dimensione dell'array

LDR R0,=V;

LDR R1, [R0];     //indirizzo primo elemento dell'array (dimensione)
LSL R1,R1, #2;   //grandezza dell'array (dimensione*4(grandezza di
una word))
LDR R4, [R0,#4]; //R4= primo elemento effettivo dell'array
MOV R2, #8;      //R2=i=contatore impostato al secondo elemento

for:
            CMP R2, R1;
            BGT fine;         //contatore>dimesione array
            LDR R3, [R0, R2]; //R3=R0[i+indirizzo primo elemento
array]
            CMP R4, R3;
            BGT then;         //R4>R3
            MOV R4, R3;       //R4=R3
then:
```

```
                ADD R2, R2, #4;  //R2=R2+4, incremento contatore di una
posizione
                B for;
      fine:
                NOP;
```

3) prototipo: word somma(word N)
    pre-condizione: N appartenente ai numeri naturale
    post-condizione: restituisce la somma dei primi N numeri naturali

```
      MOV R0, #5; //X
      MOV R1, #1; //contatore
      MOV R2, #0; //somma

      for:
                CMP R1, R0;
                BGT fine; //R0<R1, X<contatore
                ADD R2, R2, R1; //R2=R2+R1, somma=somma+contatore
                ADD R1, R1, #1;  //R1=R1+1, incremento contatore
                B for;
      fine:
                NOP;
```

4) prototipo: word Fibonacci(Word X, Word Y, word N)
    pre-condizione: N appartenente ai numeri naturale
    post-condizione: restituisce il termine N-esimo della sequenza {X, Y, X+Y, X+2Y, 2X+3Y, 3X+5Y, 5X+8Y, ...}

```
      MOV R0, #1; //X
      MOV R1, #1; //Y
      MOV R2, #5; //N
      MOV R3, #2;  //contatore, il contatore parte da 2 poichÃ© i primi
due elementi della sequanza sono x e y

      for:
                CMP R3, R2;
                BGT fine; //R2<R0, N<contatore
                MOV R4, R0;
                MOV R0, R1;
                ADD R1, R4, R1;
                ADD R3, R3, #1;
                B for;
      fine:
                NOP;  //R1= termine N-esimo
```

5) prototipo: word Fattoriale(Word X)
    pre-condizione: X appartenente ai numeri naturali
    post-condizione: restituisce il fattoriale di X

```
      MOV R0, #0; //X
      MOV R1, #1; //contatore
      MOV R2, #1; //risultato fattoriale

      for:
```

```
                CMP R1, R0;
                BGT fine; //R0<R1, N<contatore
                MUL R2, R2, R1; //R2=R2*R1
                ADD R1, R1, #1; //incremento contatore
                B for;
        fine:
                NOP;
```

6) prototipo: word swap(word[] V, word i, word j)
pre-condizione: V[0]>0 e i,j appartententi a{1, V[0]}
post-condizione: restituisce l'array V con V[i] al posto di V[j]

```
V: .word 5,3,10,8,9,1; //array, primo elemento dimensione dell'array

LDR R0,=V;

MOV R1, #2;   //R1=i
MOV R2, #5;   //R2=j

LDR R3, [R0,R1];  //R3=R0[i]=elemento in posizione i
LDR R4, [R0,R2];  //R4=R0[j]=elemento in posizione j

STR R3, [R0, R4];  //R0[j]=R3
STR R4, [R0, R3];  //R0[i]=R4
```

7) prototipo: word merge(word[] V, word[] W)
pre-condizione: V[0]>0, W[0]>0
post-condizione: restituisce un array creato unendo gli elementi di 2 array

```
V: .word 5,3,10,8,9,1;
W: .word 3,6,2,4,
Z: .word 0;

LDR R0,=V;
LDR R1,=W;
LDR R2,=Z;

LDR R3, [R0];   //primo elemento di V (dimesione v)
LDR R4, [R1];   //primo elemento di W (dimensione W)
ADD R5, R3, R4; //R5=dimensione V+dimensione W
STR R5, [R2];   //Z[0]=R5 (dimensione di Z = R5)

LSL R3, R3, #2;  //limite array V
LSL R4, R4, #2;  //limite array W
LSL R5, R5, #2;  //limite array Z

MOV R6, #4;   //contatore=i

for:
                CMP R6, R3;
                BGT finefor;       //i>dimensione V
                LDR R7, [R0, R6];  //R7=V[i]
                STR R7, [R2, R6];  //Z[i]=R7
```

```
            ADD R6, R6, #4;    //incremento contatore
            B for;
finefor:
            MOV R8, #4;      //secondo contatore = j
            ADD R6, R6, #4;  //incremento i
for2:
            CMP R8, R4;
            BGT fine;          //Rj>dimensione W
            LDR R7, [R1, R8];  //R7=W[j]
            STR R7, [R2, R6]   //Z[i]=R7
            ADD R8, R8, #4;    //incremento j
            ADD R6, R6, #4;    //incremento i
            B for2;
fine:
            NOP;
```

8) prototipo: word occorrenza(word[] V, word x)
pre-condizione: V[0]>0, x appartenente a {1, V[0]}
post-condizione: restituisce il numero di volte in cui un intero appare in un array

```
V: .word 6,6,1,1,4,9,1;

LDR R0,=V;
LDR R1, [R0];    //primo elemento di V (dimesione v)
LSL R1, R1, #2;  //limite array V

MOV R2, #1;    //R2=x=intero da ricercare
MOV R3, #4;    //contatatore=i
MOV R4, #0;    //R4=occorrenza

for:
            CMP R3, R1;
            BGT fine;           //i>R1
            LDR R5, [R0, R3];  //R5=V[i]
            CMP R5, R2;
            BEQ then;          //R5==R2
            ADD R3, R3, #4;    //incremento contatore
            B for;  //R5!=R2
then:
            ADD R4, R4, #1;  //incremento occorrenza
            ADD R3, R3, #4;  //incremento contatore
            B for
fine:
            NOP;
```

9) prototipo: word reverse(word[] V)
pre-condizione: V[0]>0
post-condizione: restituisce l'array inverito di V

```
V: .word 9,1,2,3,4,5,6,7,8,9;
W: .word 0;  //array inverao di W

LDR R0, =V;
```

```
LDR R1, =W;

LDR R2, [R0];    //primo elemento di V=dim(v)
STR R2, [R1];    //W[0]=dim(V)
LDR R3, [R1];    //primo elemento di W=dim(W)
LSL R2, R2, #2;  //ultimo indirizzo di V
LSL R3, R3, #2;  //ultimo indirizzo di W
MOV R5, R2;      //R5=ultimo indirizzo di V=j
MOV R4, #4;      //contatore=i

for:
            CMP R4, R2;
            BGT fine;            //R4>R2
            LDR R6, [R0, R5];  //R6=V[j]
            STR R6, [R1,R4];   //W[i]=R6
            ADD R4, R4, #4;    //incremento i
            SUB R5, R5, #4;    //decremento j
            B for;
fine:
            NOP;
```