

Integrazione al programma ordinamento sigle

Prima stesura del programma

Si intendono utilizzare le seguenti regole di scrittura.

- Attenti alla logica da seguire nello scrivere le istruzioni, si utilizzano dei **simboli** per indicare gli indirizzi delle locazioni di memoria, contenenti i dati, e gli indirizzi di salto. Solo in un secondo momento si sostituiranno ai simboli gli indirizzi numerici stabiliti. In particolare si ritengono le tre sigle nelle locazioni di memoria:

$v + 0$	'SUD'
$v + 1$	'CEN'
$v + 2$	'NOR'

dove al simbolo v , indirizzo della prima locazione, è assegnato il seguente significato:

$$v = 1000_8$$

'SUD', 'CEN', 'NOR' rappresentano le tre sigle in codice ASCII.

Supponendo che il numero delle sigle da esaminare (tre in questo caso) sia contenuto all'indirizzo 3000_8 , si usa il simbolo N con il seguente significato:

$$N = 3000_8$$

- Si fa inoltre l'ipotesi che il calcolatore SEC sia capace di eseguire istruzioni che sanno calcolare l'indirizzo dell'operando come somma del valore specificato nel campo operando nell'istruzione più il contenuto di una locazione di memoria di indirizzo i o j (funzione di registro indice). Così l'istruzione

LDA, 0, $1000 + i$,

con i indirizzo di una cella di memoria contenente, ad esempio il valore 6, comanda al SEC di caricare in accumulatore il dato che si trova all'indirizzo di memoria **1006**. Variando il valore contenuto nella locazione di indirizzo i , si consente alla stessa istruzione di caricare in accumulatore dati prelevati da diverse locazioni di memoria. I due indici, i e j , utilizzati in questo programma sono memorizzati nelle locazioni 3001_8 e 3002_8 , per cui viene spontaneo utilizzare altri due simboli:

$$i = 3001_8$$

$$j = 3002_8$$

per indicare gli indirizzi di queste due locazioni.

- In questa prima stesura delle istruzioni verrà omessa la cifra intermedia ottale 0.
- Per lo scambio delle stringhe, valutate in ordine non corretto, si usa la locazione di memoria di indirizzo 5000_8 . Per cui si utilizza l'ulteriore simbolo:

$$\text{temp} = 5000_8$$

N=3;

ENT	3
STA	N

i = 0;

ENT	0
STA	i

while (i < N)
{

inizio:

LDA	N
SUB	i
JPA	avanti
HLT	

j = i;

avanti:

LDA	i
STA	j

while (j < N)
{

nuovo_j:

LDA	N
SUB	j
JPA	confronta
JMP	inc_i

if (v(j) < v(i))
{

confronta:
c2:

LDA	v + i
SUB	v + j
JPA	scambia
JMP	inc_j

/confronta: diviene
/c1: nella succ. vers.

scramble(i,j);

scambia:

LDA	v + i
STA	temp
<u>s2:</u>	LDA v + j
<u>s3:</u>	STA v + i
	LDA temp
<u>s4:</u>	STA v + j

}

j++;

inc_j:

ENT	1
ADD	j
STA	j
JMP	nuovo_j

}

i++;

inc_i:

ENT	1
ADD	i
STA	i
JMP	inizio

}

Mancando nel SEC l'indirizzamento con indice le tre istruzioni con operando $v + i$ e codice operativi **LDA**, **SUB**, **STA**, non sono eseguibili.

La soluzione che si può attuare è quella di porre il valore delle tre istruzioni **LDA v**, **SUB v**, **STA v** ($v = 1000_8$) in tre locazioni distinte dal programma (di indirizzo $T1=2000_8$, $T2=2001_8$, $T3=2002_8$) e di aggiungere un gruppo di istruzioni prima dell'indirizzo "**confronta**" che, ad ogni ciclo, somma, a questi tre valori iniziali, i valori degli indici incrementati di 1 dopo ogni iterazione. Il risultato di questa somma deve essere posto nelle corrette posizioni evidenziate in giallo nel programma.

Si assume inoltre che l'indirizzo della prima istruzione del programma sia 0000_8 . Ne viene che il simbolo **inizio** assume il valore 0004_8 . Ad **avanti** il listato assegna il valore 0100_8 , a **confronta** il valore 0200_8 , a **c1** il valore 0300_8 , a **scambia** il valore 0400_8 e a **inc_j** il valore 0500_8 . Sostituendo questi valori numerici ai simboli corrispondenti nel *campo operando delle istruzioni* si ottiene il risultato riportato nelle slide.

```

ENT 3
STA N
ENT 0
STA i
inizio: LDA N
SUB i
JPA avanti
HLT
avanti: LDA i
STA j
nuovo_j: LDA N
SUB j
JPA confronta
JMP inc_i

```

/gruppo di istruzioni che aggiornano i puntatori

```

confronta: LDA T1
ADD i /uguale a LDA v + i
STA c1
STA scambia /uguale a LDA v + i
LDA T1
ADD j /uguale a LDA v + j
STA s2
LDA T3
ADD i /uguale a STA v + i
STA s3
LDA T3
ADD j /uguale a STA v + j
STA s4
LDA T2
ADD j /uguale a SUB v + j
STA c2

```

```

c1: LDA v + i /posiz. di confronta:
c2: SUB v + j /nella ver. precedente
JPA scambia
JMP inc_j
scambia: LDA v + i
STA temp
s2: LDA v + j
s3: STA v + i
LDA temp
s4: STA v + j
inc_j: ENT 1
ADD j
STA j
JMP nuovo_j
inc_i: ENT 1
ADD i
STA i
JMP inizio

```

```

v = 1000
N = 3000
i = 3001
j = 3002
temp = 5000
T1 = 2000
T2 = 2001
T3 = 2002

```

T1	LDA v
T2	SUB v
T3	STA v

