

Architettura dell'elaboratore

Riprendiamo il discorso
lasciato in sospeso
ad inizio corso ...



Prodotto tra due numeri



RAM $A * B = B + \dots + B + B \dots$

0	2000	12	19
1	7015	13	4019
2	4020	14	6007
3	2000	15	5019
4	7015	16	3000
5	4021	17	8000
6	4019	18	1
7	5020	19	0
8	1018	20	0
9	4020	21	0
10	7015	22	
11	5021	

costante per fare il decremento
risultato
A
B

Istruzioni o dati?

RAM	
...	
13	4019
14	6007
15	5019
16	3000
17	8000
18	1
19	0
20	0
21	0
22	
...	

Istruzioni

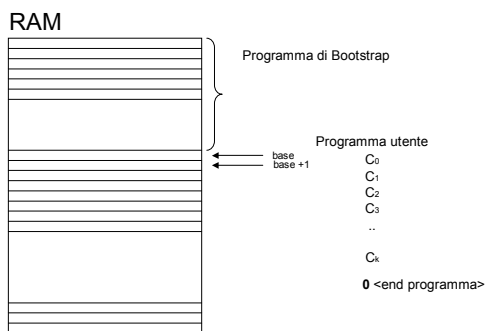
Dati

È compito del programmatore garantire che **non** si interpretino come istruzioni le celle che contengono dei dati e viceversa

Programmi automodificanti

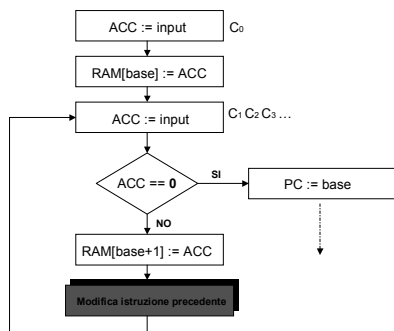
- Sono programmi che prima inseriscono un dato in una cella e poi lo interpretano come istruzione
- Sono difficili e oggi non si usano più

Il bootstrap

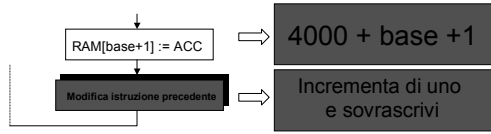


Facciamo l'ipotesi che il programma utente abbia **almeno una istruzione**

Il bootstrap: algoritmo



Il bootstrap: algoritmo



Nel seguito facciamo l'ipotesi che **base sia 20**

Il bootstrap

0	2000
1	4020
2	2000
3	7020
4	4021
5	5004
6	9
7	4004
8	6002
9	1


```

ACC := input
RAM[20] := ACC
ACC := input
if (ACC == 0) then PC := 20
RAM[21] := ACC
ACC := RAM[4]
ACC := ACC + RAM[9]
RAM[4] := ACC
PC := 2
  
```

Attenzione!
l'istruzione
RAM[4] viene
modificata in
4022

Il bootstrap

0	2000
1	4020
2	2000
3	7020
4	4022
5	5004
6	9
7	4004
8	6002
9	1

Ogni volta che si legge un nuovo input l'istruzione RAM[4] viene **incrementata**.
Così facendo, gli input C₀, C₁, ... C_k, vengono memorizzati in celle successive, a partire dall'indirizzo base (= 20)

Il bootstrap

20	C ₀
21	C ₁
22	C ₂
23	C ₃
24	C ₄
25
26
27
28	C _k
29

Dopo l'esecuzione di 7020 si passa ad eseguire l'istruzione contenuta in RAM[20], cioè la prima istruzione del programma utente appena caricato in RAM

Il bootstrap

0	2000
1	4020
2	2000
3	7020
4	40xx
5	5004
6	9
7	4004
8	6002
9	1

Problema: quando termina il caricamento del programma utente il codice del bootstrap è **diverso** da quello originale

Possiamo "resettare" l'istruzione in RAM[4] riportandola al valore iniziale **4021**

Il bootstrap

0	2000
1	4020
2	2000
3	7011
4	40xx
5	5004
6	9
7	4004
8	6002
9	1
10	4021
11	5010
12	4004
13	6020

if (ACC==0) then PC:=11

ACC := RAM[10]
RAM[4] := ACC
PC := 20

Il bootstrap (2)

Modifichiamo il programma di bootstrap considerando variabile anche il valore di **base**

Inoltre, il programma utente termina quando **C_k=8000**

Il bootstrap (2)

0	ACC := base	→	2000
1	ACC := 4000 + base	→	13
2	RAM[6] := ACC	→	4006
3	ACC := ACC + 3000	→	14
4	RAM[8] := ACC	→	4008
5	ACC := input	→	2000
6	RAM[base] := ACC	→	4000 + base
7	ACC := ACC - 8000	→	1015
8	if (ACC == 0) PC := base	→	7000 + base
9	ACC := 1	→	5016
10	ACC := ACC + RAM[6]	→	6
11	RAM[6] := ACC	→	4006
12	PC := 5	→	6005

RAM[13] := 4000
RAM[14] := 3000
RAM[15] := 8000
RAM[16] := 1

Estensione procedurale

Si definiscono dei sottoprogrammi che vengono richiamati da altri programmi per portare a termine parte del compito

Estensione procedurale

```

RAM[981] = 7995    if (ACC==0) PC:=995
RAM[982] = 4998    RAM[998]:=ACC
RAM[983] = 5999    ACC:=RAM[999]
RAM[984] = 7995    if (ACC==0) PC:=995
RAM[985] = 4997    RAM[997]:=ACC
RAM[986] = 5998    ACC:=RAM[998]
RAM[987] = 1996    ACC:=ACC-RAM[996]
RAM[988] = 7994    if (ACC==0) PC:=994
RAM[989] = 4998    RAM[998]:=ACC
RAM[990] = 5999    ACC:=RAM[999]
RAM[991] = 997     ACC:=ACC+RAM[997]
RAM[992] = 4997    RAM[997] := ACC
RAM[993] = 6986    PC:=986
RAM[994] = 5997    ACC:=RAM[997]
RAM[995] = 6000+i  PC:= i
RAM[996] = 1       costante 1
RAM[997] = 0       valore irrilevante
RAM[998] = 0       valore irrilevante
RAM[999] = 0       valore primo operando
    
```

Prodotto tra RAM[999] e il contenuto del registro ACC, risultato in ACC

Estensione procedurale

```

RAM[0] = 2000    ACC:= input
RAM[1] = 4999    RAM[999] := ACC
RAM[2] = 2000    ACC := input
RAM[3] = 6981    PC:= 981
RAM[4] = 3000    output := ACC
RAM[5] = 8000    end

Si dovrà avere RAM[995] = 6004
    
```

Le macchine convenzionali moderne ...

