Assembler

Quando si installa un programma si creano dei file che sono memorizzati nell'hard-disk del PC. Quando si "lancia" il programma parte di questi file è copiata nella RAM (memoria di lavoro) e da questo momento la CPU può eseguire le istruzioni del programma.

Per capire meglio cos'è esecuzione di una serie di istruzioni in una shell MSDOS lanciamo il programma debug che ci mette a disposizione un emulatore di PC con gli strumenti per analizzarne il funzionamento.

Il programma ha solo tre istruzioni, la terza altera la prima. In assembler la CPU fa esattamente quello che le istruzioni indicano, anche modificare una parte del programma stesso.

```
C:\Users\claudio>debug
-a 100
17C6:0100 mov ax,1234
17C6:0103 mov bx,0100
17C6:0106 mov [bx],ax
17C6:0108
-r
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17C6  ES=17C6  SS=17C6  CS=17C6  IP=0100   NV UP EI PL NZ NA PO NC
17C6:0100 B83412        MOV     AX,1234
-t=100, 1

AX=1234  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17C6  ES=17C6  SS=17C6  CS=17C6  IP=0103   NV UP EI PL NZ NA PO NC
17C6:0103 BB0001        MOV     BX,0100
-t 1

AX=1234  BX=0100  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17C6  ES=17C6  SS=17C6  CS=17C6  IP=0106   NV UP EI PL NZ NA PO NC
17C6:0106 8907          MOV     [BX],AX                         DS:0100=34B8
-d 100,107
17C6:0100  B8 34 12 BB 00 01 89 07                      .4......
-t 1

AX=1234  BX=0100  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17C6  ES=17C6  SS=17C6  CS=17C6  IP=0108   NV UP EI PL NZ NA PO NC
17C6:0108 0000          ADD     [BX+SI],AL                      DS:0100=34
-d 100,107
17C6:0100  34 12 12 BB 00 01 89 07                      4.......
-q
C:\Users\claudio>
```

In verde sono evidenziati i comandi del debug

`a 100`
Attiva la scrittura di istruzioni assembler nella locazione 0100.

`17C6:0100 mov ax,1234`
Sposta il valore 1234 nel registro AX

`17C6:0103 mov bx,0100`
Sposta il valore 0100 nel registro BX

`17C6:0106 mov [bx],ax`
Sposta il valore nel registro AX nella locazione puntata da BX

`-r`
Visualizza i registri della CPU,
AX contiene 0000,

BX contiene 0000,
IP contiene 0100 che punta all'istruzione MOV AX,1234.

```
-t=100, 1
```

Esegue un istruzione contenuta a partire dalla locazione 0100 e successivamente visualizza lo stato dei registri della CPU,
AX contiene 1234,
IP contiene 0103 e punta all'istruzione MOV BX,0100

```
-t 1
```

Esegue un istruzione contenuta a partire dalla locazione puntata da IP e successivamente visualizza lo stato dei registri della CPU,
BX contiene 0100,
IP contiene 0106 e punta all'istruzione MOV [BX],AX

```
-t 1
```

Esegue un istruzione contenuta a partire dalla locazione puntata da IP e successivamente visualizza lo stato dei registri della CPU,
la locazione 0100 contiene 34,
la locazione 0101 contiene 12.

Il programma di tre istruzioni occupa 8 byte. Ogni istruzione è suddivisa in codice operativo e dato. Il codice operativo è sempre presente, il dato può non esserci.

indirizzo	valore	informazione	istruzione	descrizione
0100	B8	Codice operativo	MOV AX,1234	Indirizzamento immediato il valore 1234 è spostato nel registro AX
0101	34	Dato		
0102	12	Dato		
0103	BB	Codice operativo	MOV BX,0100	Indirizzamento immediato il valore 0100 è spostato nel registro BX
0104	00	Dato		
0105	01	Dato		
0106	89	Codice operativo	MOV [BX],AX	Indirizzamento indiretto AX è spostato nelle locazioni puntate da BX
0107	07	Codice operativo		

Con il termine **fetch** si intende l'operazione di caricamento dell'istruzione dalla RAM alla CPU che poi la decodifica e la esegue nella fase di **execute**.

Si può suddividere l'esecuzione delle tre istruzioni in una serie di letture e scritture in RAM.

address	AX	BX	IP	data	Registro istruzione	fase	R/W	0100	0101
0100	0000	0000	0100	B8	B8	fetch	R	B8	34
0101	0000	0000	0101	34	B8 34	fetch	R	B8	34
0102	0000	0000	0102	12	B8 34 12	fetch	R	B8	34
	1234	0000	0103		B8 34 12	execute		B8	34
0103	1234	0000	0103	BB	BB	fetch	R	B8	34
0104	1234	0000	0104	00	BB 00	fetch	R	B8	34
0105	1234	0000	0105	01	BB 00 01	fetch	R	B8	34
	1234	0100	0106		BB 00 01	execute		B8	34
0106	1234	0100	0106	89	89	fetch	R	B8	34
0107	1234	0100	0107	07	89 07	fetch	R	B8	34
0100	1234	0100	0108	34	89 07	execute	W	34	34
0101	1234	0100	0108	12	89 07	execute	W	34	12

Questa descrizione è semplificata rispetto al funzionamento di una CPU attuale, si tratta della tecnologia usata nelle CPU dei primi anni ’80. Per aumentare la velocità di esecuzione sono state introdotte diverse tecniche.

Prefetch. Mentre è ancora in esecuzione l’istruzione si procede al fetch dell’istruzione successiva.
Branch prediction. Aumentando la dimensione della memoria dedicata al prefetch si pone il problema di come gestire le istruzioni condizionali. Sono stati studiati sofisticati algoritmi per gestire al meglio queste situazioni.
Dual channel. Dopo aver aumentato la dimensione del bus da 8 a 16 e 32 bit il data bus per velocizzare il sistema si sono affiancati due moduli per ottenere un bus a 64 bit.

I **compilatori** (C, C++, Pascal, Basic) trasformano un file che contiene il codice sergente in un file che contiene il codice macchina che è una serie di istruzioni tipo quelle viste in precedenza. Il codice macchina è il sistema più efficiente per sfruttare la potenza del PC. L’inconveniente principale è l’accesso diretto della CPU alla memoria (Nell’esempio precedente il programma altera se stesso). Altro inconveniente è il fatto che al cambio di ogni CPU si deve procedere all’ottimizzazione del codice macchina con una nuova compilazione.

Gli **interpreti** (Java, .net) partendo dal codice sorgente non generano codice macchina ma un codice intermedio identico per tutte le piattaforme. Ogni piattaforma ha una propria versione dell’interprete in grado di eseguire il codice intermedio senza nessuna modifica. L’inserimento di un livello ulteriore consente una gestione più sofisticata della sicurezza evitando che processi distinti interferiscano tra di loro.

```
-a 200
0B1A:0200 mov dx,0378
0B1A:0203 in al,dx
0B1A:0204 mov dx,0379
0B1A:0207 out dx,al
0B1A:0208
-u 200,207
0B1A:0200 BA7803        MOV     DX,0378
0B1A:0203 EC            IN      AL,DX
0B1A:0204 BA7903        MOV     DX,0379
0B1A:0207 EE            OUT     DX,AL
```

indirizzo	valore	informazione	istruzione	descrizione
0200	BA	Codice operativo	MOV DX,0378	Indirizzamento immediato il valore 0378 è copiato nel registro DX
0201	78	Dato		
0202	03	Dato		
0203	EC	Codice operativo	IN AL,DX	Il contenuto della porta puntata da DX è copiato nel registro AL
0204	BA	Codice operativo	MOV DX,0379	Indirizzamento immediato il valore 0379 è copiato nel registro DX
0205	79	Dato		
0206	03	Dato		
0207	EE	Codice operativo	OUT DX,AL	Il contenuto di AL è copiato nella porta puntata da DX

address	AX	DX	IP	data	Registro istruzione	fase	R/W	Porta 0378	Porta 0379
0200	0000	0000	0200	BA	BA	fetch	R	AA	BB
0201	0000	0000	0201	78	BA 78	fetch	R	AA	BB
0202	0000	0000	0202	03	BA 78 03	fetch	R	AA	BB
	0000	0378	0203		BA 78 03	execute		AA	BB
0203	0000	0378	0203	EC	EC	fetch	R	AA	BB
0378	00AA	0378	0204	AA	EC	execute	R	AA	BB
0204	00AA	0378	0204	BA	BA	fetch	R	AA	BB
0205	00AA	0378	0205	78	BA 79	fetch	R	AA	BB
0206	00AA	0378	0206	03	BA 79 03	fetch	R	AA	BB
	00AA	0379	0207		BA 79 03	execute		AA	BB
0207	00AA	0379	0207	EE	EE	fetch	R	AA	BB
0379	00AA	0379	0208	AA	EE	execute	W	AA	AA

```
-a 300
0B1A:0300 mov dx,0379
0B1A:0303 in al,dx
0B1A:0304 mov bx,1000
0B1A:0307 mov [bx], al
0B1A:0309
-u 300,308
0B1A:0300 BA7903        MOV     DX,0379
0B1A:0303 EC            IN      AL,DX
0B1A:0304 BB0010        MOV     BX,1000
0B1A:0307 8807          MOV     [BX],AL
-
```

indirizzo	valore	informazione	istruzione	Descrizione
0300	BA	Codice operativo	MOV DX,0379	Indirizzamento immediato il valore 0379 è copiato nel registro DX
0301	79	Dato		
0302	03	Dato		
0303	EC	Codice operativo	IN AL,DX	Il contenuto della porta puntata da DX è copiato nel registro AL
0304	BB	Codice operativo	MOV BX,0100	Indirizzamento immediato il valore 0100 è copiato nel registro BX
0305	00	Dato		
0306	10	Dato		
0307	88	Codice operativo	MOV [BX],AL	Il contenuto di AL è copiato nella locazione puntata da BX
0308	07	Codice operativo		

address	AX	BX	DX	IP	data	Registro istruzione	fase	R/W	Porta 0379	Locazione 1000
0300	0000	0000	0000	0300	BA	BA	fetch	R	AA	BB
0301	0000	0000	0000	0301	79	BA 79	fetch	R	AA	BB
0302	0000	0000	0000	0302	03	BA 79 03	fetch	R	AA	BB
	0000	0000	0379	0303		BA 79 03	execute		AA	BB
0303	0000	0000	0379	0303	EC	EC	fetch	R	AA	BB
0379	00AA	0000	0379	0304	AA	EC	execute	R	AA	BB
0304	00AA	0000	0379	0304	BB	BB	fetch	R	AA	BB
0305	00AA	0000	0379	0305	00	BB 00	fetch	R	AA	BB
0306	00AA	0000	0379	0306	10	BB 00 10	fetch	R	AA	BB
	00AA	1000	0379	0307		BB 00 10	execute		AA	BB
0307	00AA	1000	0379	0307	88	88	fetch	R	AA	BB
0308	00AA	1000	0379	0307	07	88 07	fetch	R	AA	BB
1000	00AA	1000	0379	0308	AA	88 07	execute	W	AA	AA

```
-a 400
0B1A:0400 mov bx,3000
0B1A:0403 mov al,[bx]
0B1A:0405 mov bx,4000
0B1A:0408 mov [bx],al
0B1A:040A
-u 400,409
0B1A:0400 BB0030        MOV     BX,3000
0B1A:0403 8A07          MOV     AL,[BX]
0B1A:0405 BB0040        MOV     BX,4000
0B1A:0408 8807          MOV     [BX],AL
-
```

indirizzo	valore	informazione	istruzione	descrizione
0400	BB	Codice operativo	MOV BX,3000	Indirizzamento immediato il valore 3000 è copiato nel registro BX
0401	00	Dato		
0402	30	Dato		
0403	8A	Codice operativo	MOV AL,[BX]	Il contenuto della locazione puntata da BX è copiato nel registro AL
0404	07	Codice operativo		
0405	BB	Codice operativo	MOV BX,4000	Indirizzamento immediato il valore 4000 è copiato nel registro BX
0406	00	Dato		
0407	40	Dato		
0408	88	Codice operativo	MOV [BX], AL	Il contenuto di AL è copiato nella locazione puntata da BX
0409	07	Codice operativo		

address	AX	BX	IP	data	Registro istruzione	fase	R/W	Locazione 3000	Locazione 4000
0400	0000	0000	0400	BB	BB	fetch	R	AA	BB
0401	0000	0000	0401	00	BB 00	fetch	R	AA	BB
0402	0000	0000	0402	30	BB 00 30	fetch	R	AA	BB
	0000	3000	0403		BB 00 30	execute		AA	BB
0403	0000	3000	0403	8A	8A	fetch	R	AA	BB
0404	0000	3000	0404	07	8A 07	fetch	R	AA	BB
3000	00AA	3000	0405	AA	8A 07	execute	R	AA	BB
0405	00AA	3000	0405	BB	BB	fetch	R	AA	BB
0406	00AA	3000	0406	00	BB 00	fetch	R	AA	BB
0407	00AA	3000	0407	40	BB 00 40	fetch	R	AA	BB
	00AA	4000	0408		BB 00 40	execute		AA	BB
0408	00AA	4000	0408	88	88	fetch	R	AA	BB
0409	00AA	4000	0409	07	88 07	fetch	R	AA	BB
4000	00AA	4000	040A	AA	88 07	execute	W	AA	AA