

Primo esempio (tutte le celle di memoria contengono 0)

```
0 MOV EAX,0
1 MOV EBX,0
2 addition:
3 ADD EAX,[ECX]
4 INC EBX
5 ADD ECX,4 ; we add four to read every fourth byte
6 CMP EBX,4
7 JNE addition ; EAX now contains the expected value
8
```

Help	Effects: DEST := SRC; Copies the source value (second argument) to the destination (first argument).
7-Segment	
StripLight	Flags to be set: none
Console	Misc: Both arguments must be the same size (byte, word or doubleword). Move cannot be used to load the CS register, use JMP, CALL or RET instead.
Graphics	
StripLight	Flags to be set: none
Console	Misc: Both arguments must be the same size (byte, word or doubleword). Move cannot be used to load the CS register, use JMP, CALL or RET instead.
Graphics	

Esecuzione passo-passo: 

Registers	
	bin ±dec dec hex
EAX:	0
EBX:	1

```
0 MOV EAX,0
1 MOV EBX,0
2 addition:
3 ADD EAX,[ECX]
4 INC EBX
5 ADD ECX,4 ; we add four to read every fourth byte
6 CMP EBX,4
7 JNE addition
```

Ha incrementato il contenuto di EBX e punta l'istruzione successiva

Registers	
	bin ±dec dec hex
EAX:	0
EBX:	1
ECX:	4

```
0 MOV EAX,0
1 MOV EBX,0
2 addition:
3 ADD EAX,[ECX]
4 INC EBX
5 ADD ECX,4 ; we add four to read every fourth byte
6 CMP EBX,4
7 JNE addition
```

Ha sommato 4 al contenuto di ECX

Solo dopo 4 incrementi del contenuto di EBX non salta più all'etichetta **addition**

Registers window:

bin	±dec	dec	hex
EAX:	0		
EBX:	4		
ECX:	16		
EDX:	0		
ESI:	0		
EDI:	0		
ESP:	4096		
EBP:	4096		
EIP:	7		

Assembly window:

```

0 MOV EAX,0
1 MOV EBX,0
2 addition:
3 ADD EAX,[ECX]
4 INC EBX
5 ADD ECX,4 ; we add four to read every fourth byte
6 CMP EBX,4
7 JNE addition
8 EBX now contains the expected value
9
10
11
12
13
14
15
16
17
18
19
20
21
    
```

<input type="checkbox"/> Carry	<input type="checkbox"/> Overflow
<input type="checkbox"/> Sign	<input checked="" type="checkbox"/> Zero
<input checked="" type="checkbox"/> Parity	<input checked="" type="checkbox"/> Auxiliary
<input type="checkbox"/> Trap	<input type="checkbox"/> Direction

ed il confronto con 4 setta i flag Zero e Parity (oltre ad Auxiliary cioè **Half-carry flag** o **Decimal adjust flag**)

Primo esempio (nella cella di memoria con indirizzo 0 si inserisce 2)

Memory			
desc	hex	highlight	
			8 Bit 16Bit 32E
address	signed int	unsigned int	hex
0x0	2	2	0x00000002
0x4	0	0	0x00000000
0x8	0	0	0x00000000

Allora: sommando il contenuto puntato da ECX (cioè 2) ad al contenuto di EAX (cioè 0) si ottiene in EAX 2

Registers window:

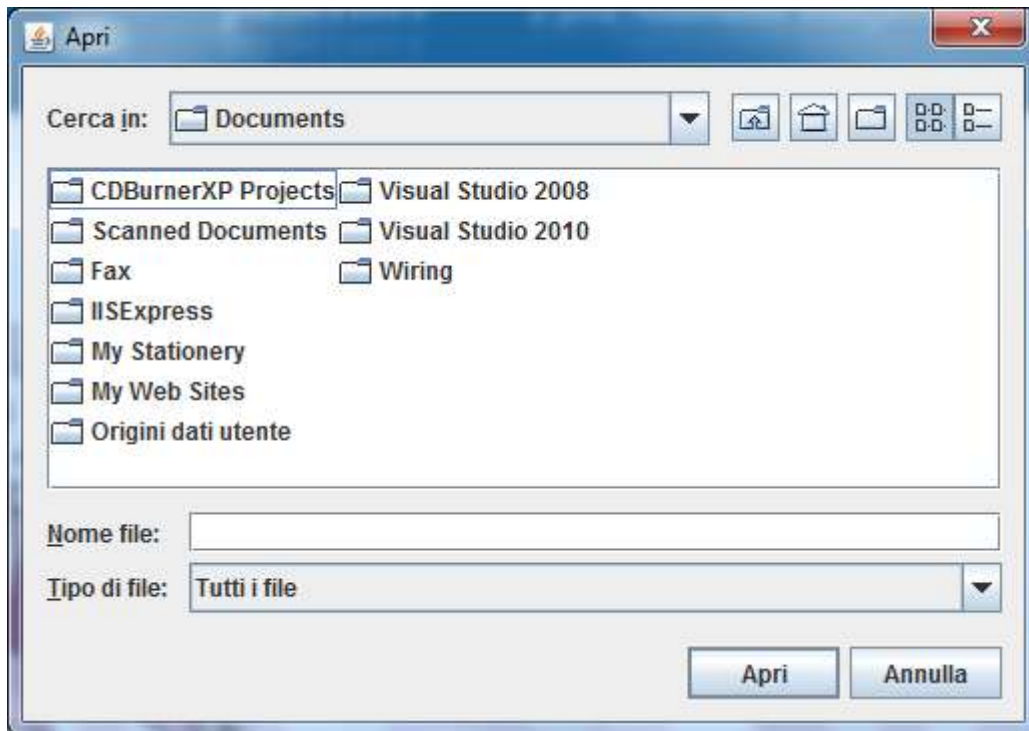
bin	±dec	dec	hex
EAX:	2		

Assembly window:

```

0 MOV EAX,0
1 MOV EBX,0
2 addition:
3 ADD EAX,[ECX]
4 INC EBX
    
```

Salvataggio di default nella cartella Documents:



```
; first we set EAX and EBX to zero
```

```
MOV EAX, 0
```

```
MOV ECX, 0
```

```
; then we create a loop that performs the addition
```

```
addition:
```

```
; note the addition of 4 in each iteration, because the address is given  
in bytes
```

```
ADD EAX, [EDX] ; somma il contenuto indirizzato da EDX al contenuto di EAX
```

```
INC ECX
```

```
ADD EDX, 4
```

```
CMP ECX 4
```

```
JNE addition
```

Il linguaggio **assembly**, detto anche linguaggio assembleativo, è, tra i linguaggi di programmazione, quello più vicino al linguaggio macchina vero e proprio (*binario*), pur essendo differente rispetto a quest'ultimo.

Erroneamente viene spesso chiamato "*assembler*", anche se quest'ultimo termine identifica il programma "*assemblatore*" che converte il linguaggio assembly in linguaggio macchina.

Documentazione → <http://wwwi10.lrr.in.tum.de/~jasmin/documentation.html>
→ <http://wwwi10.lrr.in.tum.de/~jasmin/basics.html> (istruzioni)