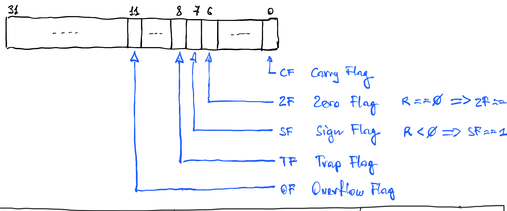
CONDITION CODE (cc)	PROPRIETÀ DI R
e / z	R == Ø
ne / nz	R != Ø
g / l / ge / le	R > Ø R < Ø R ≥ Ø R ≤ Ø
a / b / ae / be	R > Ø R < Ø R ≥ Ø R ≤ Ø

REGISTRO EFLAGS o STATUS REGISTER

E' un registro con bit (flag) dedicati a rappresentare lo stato di funzionamento del programma nella cpu



CODICE C ORIGINALE	CODICE C EQ	CODICE ASM

if (Test) {
  blocco A
}
blocco B
	if (!test) goto E
	blocco A
	E:
	blocco B
		cmp S, D
		jcc E
		blocco A
		E:
		blocco B

CODICE C ORIGINALE	CODICE C EQ	CODICE ASM

if (Test) {
  blocco 1
} else {
  blocco 2
}
blocco 3
	if (!test) goto E
	  blocco 1
	  goto F
	E:
	  blocco 2
	F:
	blocco 3
		cmp S, D
		jcc E
		blocco 1
		jmp F
		E:
		blocco 2
		F:
		blocco 3

ISTRUZIONE TEST

CMP S, D	D - S
CMP Ø, D	D - Ø
TEST D, D	D & D

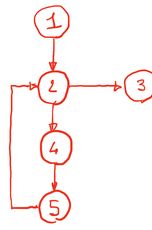
CICLO WHILE IN ASM

CODICE C	CODICE C EQ 1	CODICE C EQ 2	CODICE ASM

while (Test) {
  blocco 1
}
blocco 2
	L:
	if (test) {
	  blocco 1
	  goto L;
	}
	blocco 2
		L:
		if (!test) goto E;
		blocco 1
		goto L;
		E:
		blocco 2
			L:
			cmp S, D
			jcc E
			blocco 1
			jmp L
			E:
			blocco 2

CICLO FOR IN ASM

```
for ( init ; test ; update ) {
      blocco 1
}
blocco 2
```



CODICE C	CODICE EQ 1

for (init ; test ; update) {
  blocco 1
}
blocco 2
	init;
	while (Test) {
	  blocco 1
	  update;
	}
	blocco 2

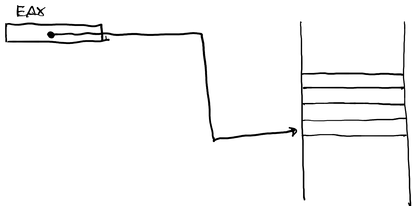
GESTIONE PUNTATORI IN ASM

Un puntatore in C di Tipo T* contiene l'indirizzo di memoria dove è salvato un dato di Tipo T

```
int * a
:
*a = Ø;
```

```
# int *a → eax
:
movl $Ø, (%eax)
```

IMPORTANTE: se uso (reg), reg deve contenere un indirizzo di memoria valido



Quando opero con i puntatori devo sempre usare 2 passaggi:

1. Salvo l'indirizzo in un registro
2. Accedo all'indirizzo contenuto nel registro

IMPORTANTE: per una istruzione OP con due operandi UNO al massimo può essere un accesso in memoria