C e Assembly

Assembler in line

```
  char vett[20];
int i;
i=1;
 asm {
mov ax,byte ptr vett[i]
}
Su:
 asm {
…..
…..
Loop su
}
```

Assemblaggio separato

File in assembly

```
Dseg segment
_varcom db 54
Dseg ends

Cseg segment
 Public _nome:far,_varcom:byte
_nome proc far.
……
…….
…
ret
_nome endp
```

File in C

```
include <...>
extern char varcom;
extern "C" { void far nome(a,b); }
main()
….
….
varcom=34;
nome(13,56)
```

Valori restituiti in …

Supponendo che il file in C si chiami pippo.cpp e quello in assembly beta.asm e volendo generare un eseguibile di nome pluto, dovremo digitare :

```
bcc -v -ms -epluto pippo.cpp beta.asm
```

Accesso alle porte

1. outport(numero di porta,valore)

In Assembler

```
Mov al,4bh
Out 32h,al
```

In C

```
Outport(0x32,0x4b)
```

2. Destinazione= inport (numero di porta)

In Assembler

```
in al,4bh
mov n,al
```

In C

```
N=inport(0x4b)
```

Esempio

In assembler

```
In al,60h
Test al,80h
Jnz esci
In al,61h
And al,0feh
Xor al,02h
Out 61h,al
```

In C

```
If (!(inportb(0x60)&0x80))
Outportb(0x61,(inport(0x61)&0xFE)^2)
```

Accesso tramitre i registri ai servizi BIOS e DOS

Nel file di intestazione <dos.h> è definita una variabile di tipo union regs mediante la quale si può accedere direttamente ai registri della CPU
[La union è un tipo di dati derivato, simile alla struct , ma particolare perché quando viene dichiarata alloca in memoria lo stesso spazio per tutte le sue componenti, spazio che è pari alla componente più ingombrante

L' esempio :

```
union {
Int a;
char b;
} N;
```

definisce una variabile di tipo union che occuperà 2 byte ;questo vuol dire che potrà contenere sia una variabile di tipo char che avrà nome N.b, sia una di tipo integer che si chiamerà N.a.
Non posso dichiarare contemporaneamente le due variabili nella stessa union perchè i 2 dati occupano lo stesso spazio in memoria.]
La union definita nel file di intestazione <dos.h> è la seguente:

```
union REGS
 {  struct WORDREGS x;
   Struct BYTEREGS  h;
};
 Dove

struct WORDREGS
  { unsigned int AX,BX,CX,DX;
    Unsigned int SI,DI,CFLAG,FLAG; dove CFLAG sta per il flag di carry
};

struct BYTEREGS
  { unsigned char AL,AH,BL,BH;
    Unsigned char CL,CH,DL,DH;
```

};

Le funzioni del C che usano la union regs per richiamre servizidi sistema sono

La funzione int86 che permette di richiamare un'interruzione software qualsiasi
La sintassi è la seguente:

int86(interruzione, union di input, union di output)

in assembly	**in C**
mov ah,5	union regs rin,rout;
mov cl,codscan	char c;
mov ch,codascii	rin.h.ah=0x05;
int 16h	rin.h.cl=0x1f;
	rin.h.ch='a';
in AL viene restituito il codice d'errore	int86(0x16,&rin,&rout);
	c =rout.h.al

alcune funzioni invece servono solo per richiamre servizi dell'INT 21H del DOS e sono

1. **intdos**(union in, union out)
2. **bdos**(servizio, valore per DX, Valori per AL) quest'ultima serve solo per quei servizi che usano solo i registri dx e al

La funzione **getvect**(tipo n) viene usata per reperire il vettore di interrupt del valore passato come parametro; restituisce un puntatore far .Per il salvataggio di questo **puntatore** si deve definire una variabile di tipo **interrupt** con la seguente definizione
void interrupt (*oldIrq)(void);
ad esempio se vogliamo salvare il vettore di interrupt della tastiera dobbiamo scrivere

```
void interrupt (*oldint)(void);

oldint= getvect(9)
```

la funzione **setvect(**int n, nuovaint) permette di salvare il nuovo vettore di interrupt nella posizione indicata dal tipo

Ad esempio

```
setvect(9,mioint)
```

sostituisce il puntatore della routine standard della tastiera con quella della procedura mioint che deve essere dichiarata nel modo seguente:

```
void interrupt mioint(void)
```