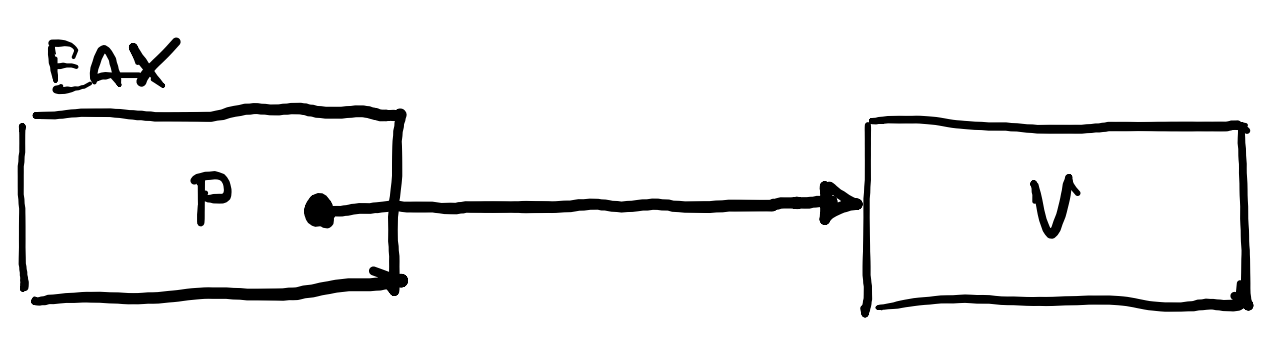
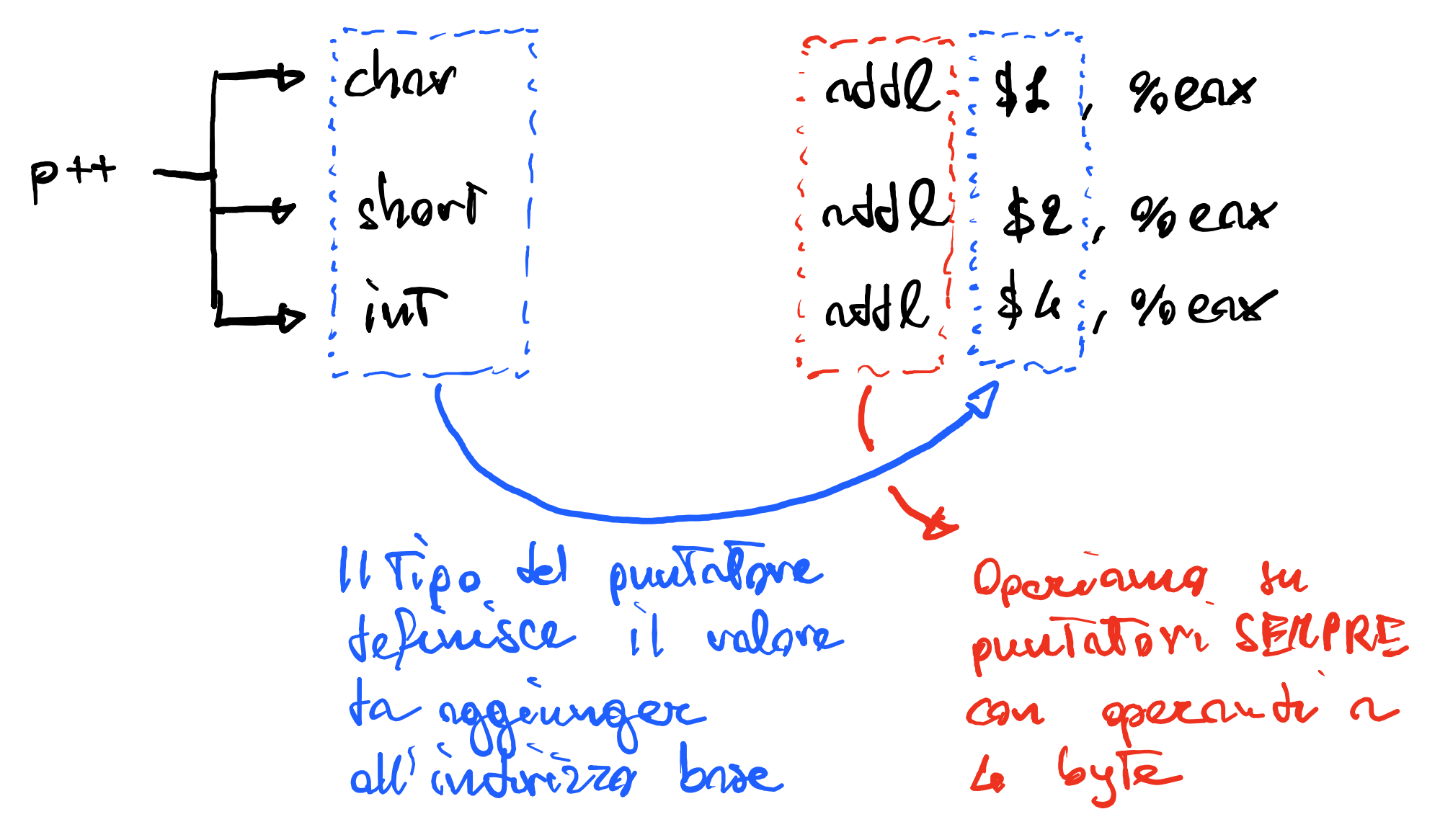
SC 17/3/2026

17/03/26 09:03

ARITMETICA DEI PUNTATORI



CODICE C



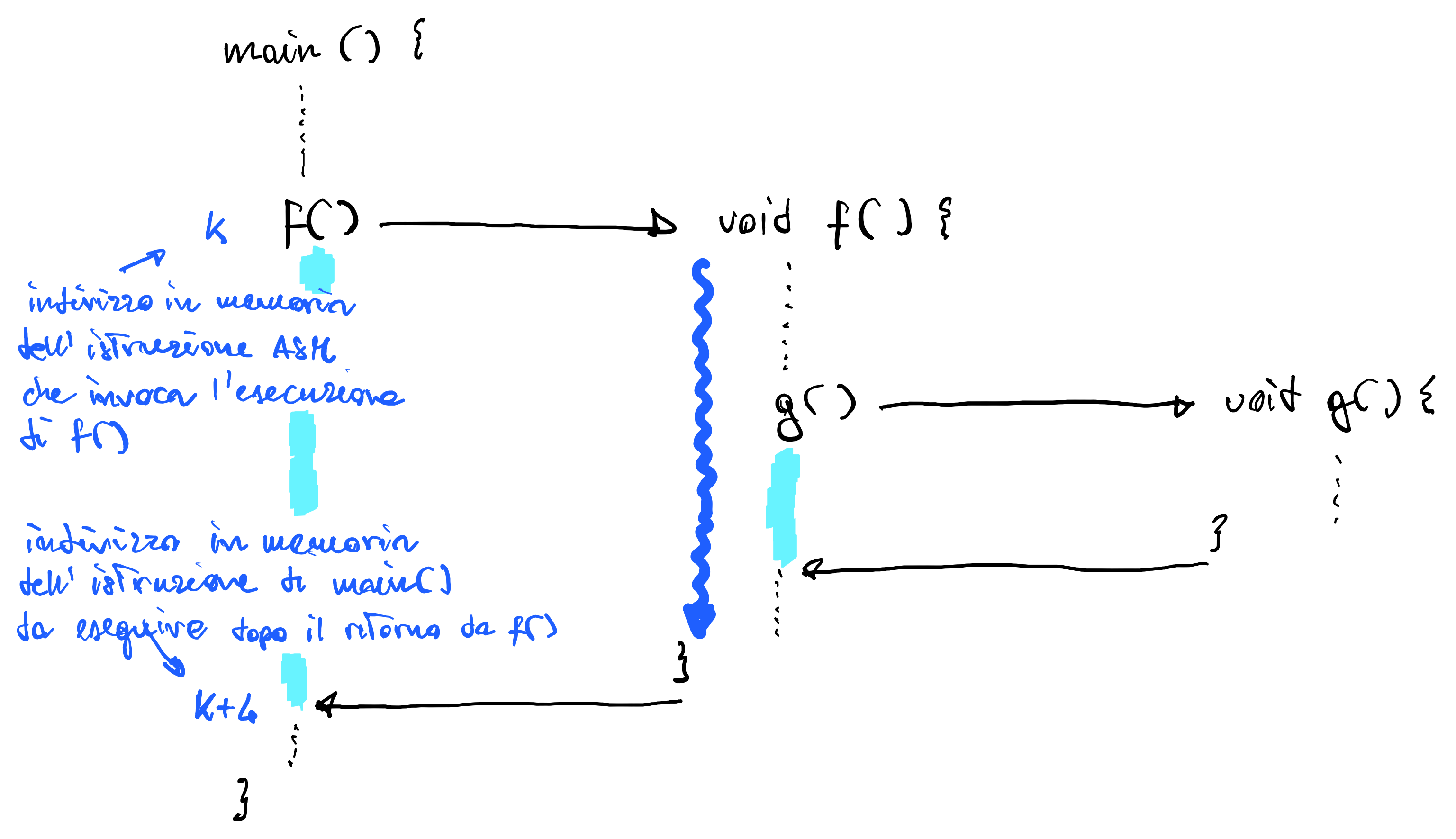
Il Tipo del puntatore definisce il valore da aggiungere all'indirizzo base

Operiamo su puntatori SEMPRE con operandi a 4 byte

ARITMETICA DEI PUNTATORI <u>IMPLICITA</u>

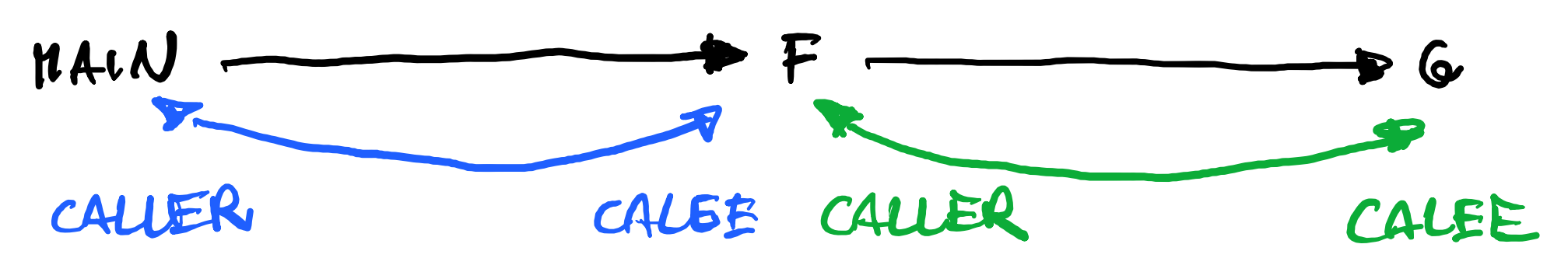
ARITMETICA DEI PUNTATORI <u>ESPLICITA</u>

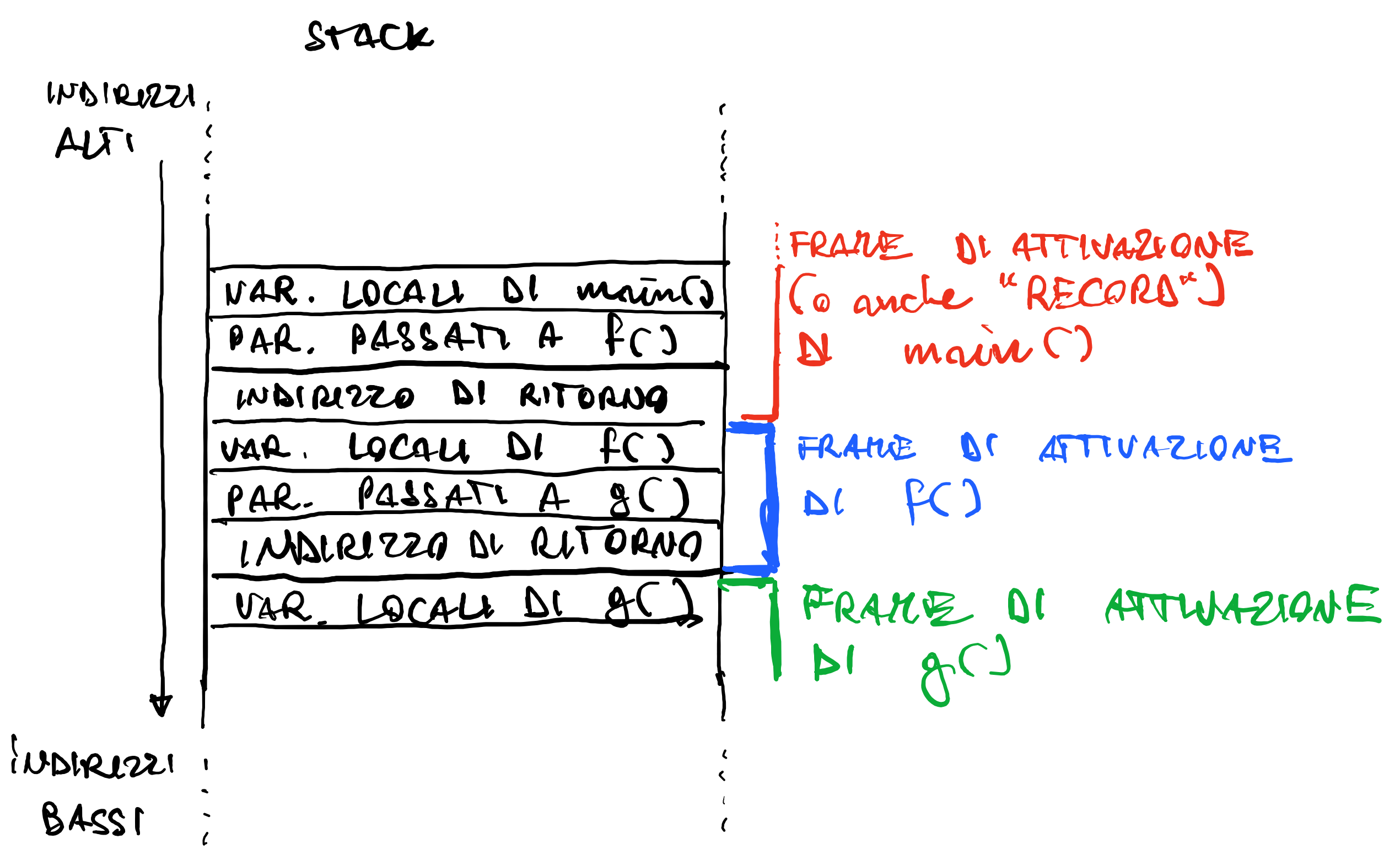
CHIAMATE A FUNZIONE



indirizzo in memoria dell'istruzione ASM che invoca l'esecuzione di f()

indirizzo in memoria dell'istruzione di main() da eseguire dopo il ritorno da f()





Cosa contiene il frame/record di attivazione di una funzione:

① Variabili locali

② copie dei registri (ebx, edi, esi, ebp)

③ parametri per le funzioni chiamate

④ indirizzo di ritorno dalle funzioni chiamate

COSA ACCADE QUANDO CHIAMO UNA FUNZIONE

```
int f() {
    int x = g();      call g
    x=x+1;
    return x;
}
```

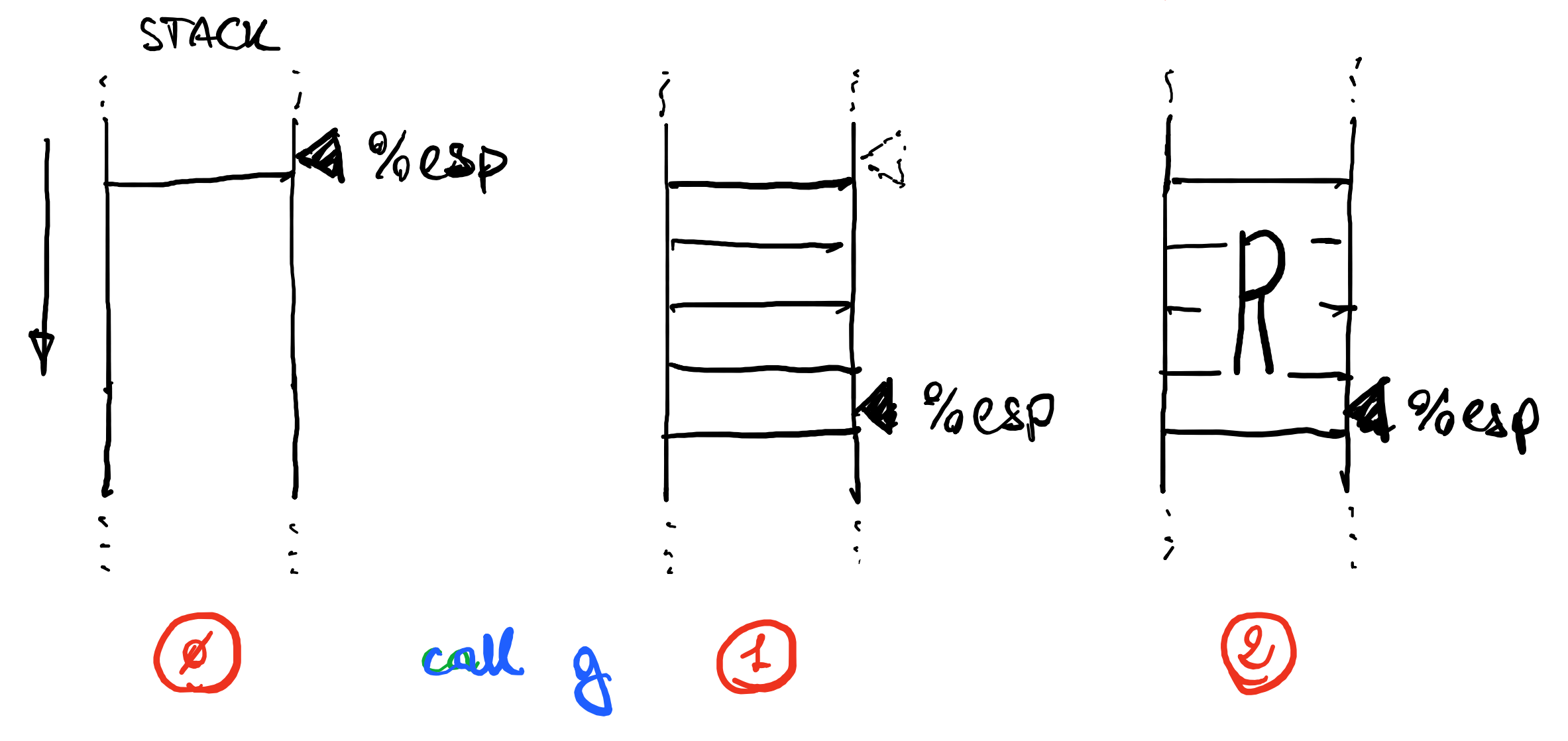
ESECUZIONE DI CHIAMATA A FUNZIONE →

① vengono riservati nella stack 4 byte come se fosse eseguita

subl $4, %esp

② viene scritto nei 4 byte appena riservati l'indirizzo di ritorno R per la funzione chiamata

③ %eip viene sovrascritto con l'indirizzo della prima istruzione della funzione chiamata



COSA ACCADE QUANDO ESEGUO RET

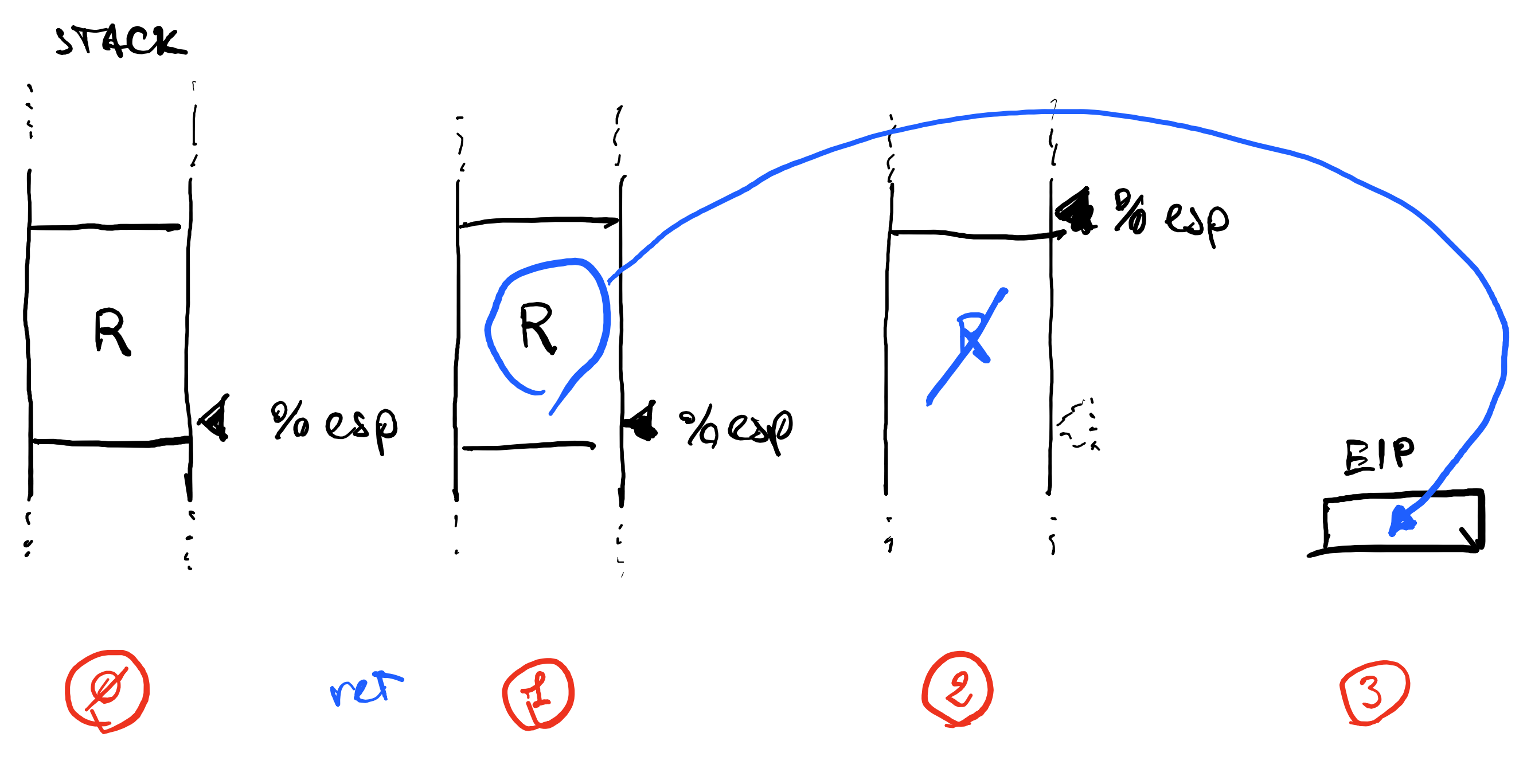
ret RITORNO ALLA FUNZIONE CHIAMANTE →

① legge 4 byte da %esp → R

② libera 4 byte dalla stack come se fosse eseguita

addl $4, %esp

③ scrive R in %eip

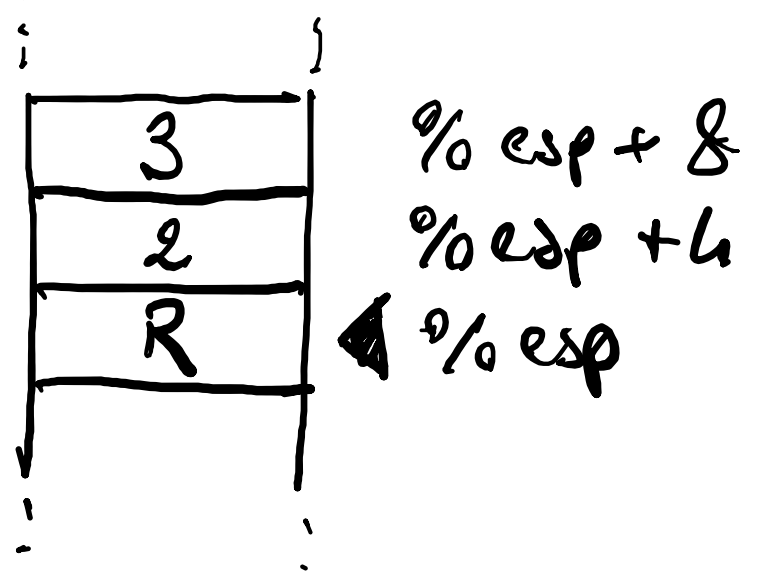


IMPORTANTE: Prima di chiamare "ret" il valore di %esp DEVE essere uguale a quello che aveva all'inizio dell'esecuzione della funzione

CHIAMATA A FUNZIONE CON ARGOMENTI

```
int f() {
    int x = g(2,3);
    x++;
    return x;
}
```

quando chiamo g(...) cosa ci aspetta di trovare nella stack?

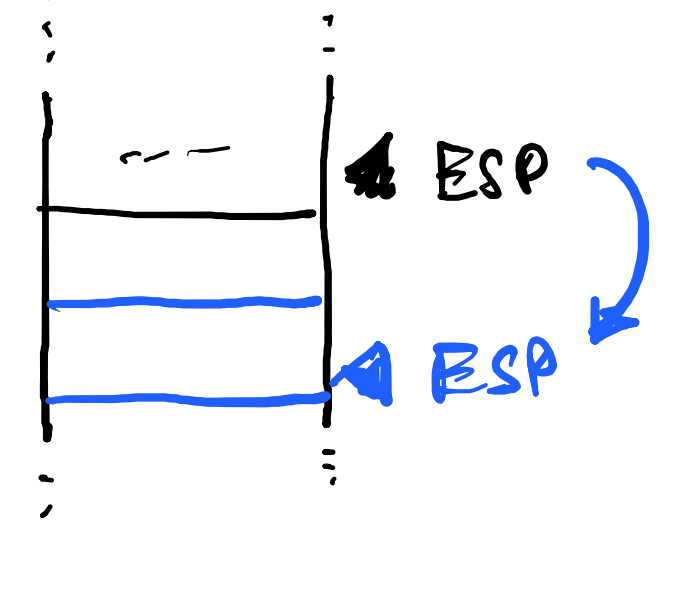


PREPARAZIONE STACK CON PARAMETRI

① RISERVARE SPAZIO PER I PARAMETRI

subl $8, %esp

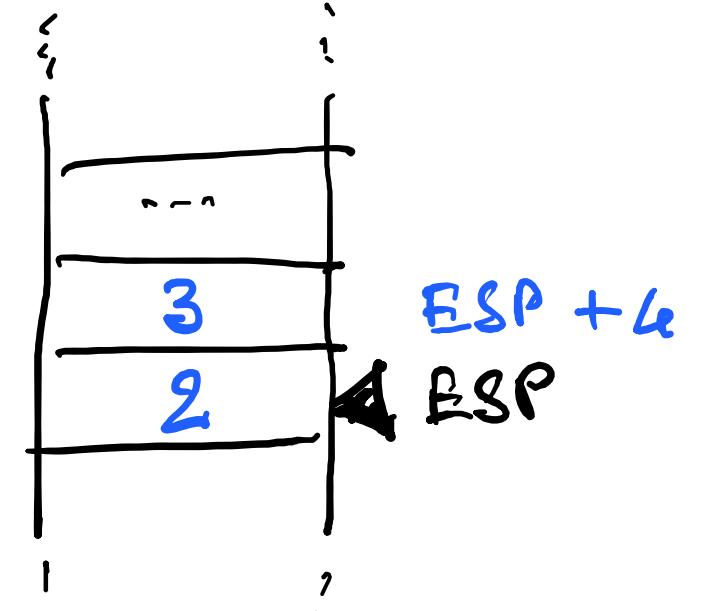
4 * num. parametri



② COPIARE I VALORI ATTUALI DEI PARAMETRI NELLO SPAZIO RISERVATO IN ORDINE INVERSO

movl $3, 4(%esp)

movl $2, (%esp)



③ CHIAMATA A FUNZIONE

call g