martedì 24 ottobre 2017 16:09

while (e) { corpo } ;

do { corpo } while (e) ;

compilazione (while (e) { corpo }) =

"loop" IF$_{NE}$ → "fineciclo"
compilazione (corpo)
GOTO "loop"
fineciclo : ---

⇓

loop IF$_{NC}$ fine
{corpo}
IF$_{NC}$ fine ←
{corpo}
GOTO loop ←
fine ---

while (true) {
× preleva istr @ IC
× decodifica
esegue { (IC ← IC+1)
salto IC ← ----)
halt interruzione
}

for ( ) { body }

init
loop : { body } ←
update (i)
test (i)

for (int i=0 ; i<N ; i++)
x[i] = b[i] ;

loop : ADD R_0, R_0, R_i
LOAD $R_{base\,b}$, R_i, R_1
STORE $R_{base\,x}$, R_i, R_1
INCR R_i
IF$_<$ R_i, R_N, loop
end : —

N pari

for (int i=0; i<N; i+=2) {
x[i] = b[i];
x[i+1] = b[i+1];
}

|||

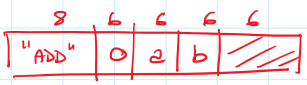
unrolling

loop : LOAD ←
STORE
INCR
LOAD
STORE
INCR
IF$_<$

istruzioni "derivate"

$i = \phi$ ADD R_ϕ, R_ϕ, R_i ≡ CLEAR R_i

ADD R_ϕ, R_a, R_b ≡ MOV R_a, R_b REG[a] → REG[b]



cod op ≠ { LOAD $R_{base}, R_{indice}, R_{dest}$; LOAD $R_{base}, \#m, R_{dest}$ }

chiamata di procedura/funzione

CALL R_f, R_{ret} ← x contenere l'indirizzo di ritorno

↑ indirizzo della funzione o procedura che vogliamo chiamare

R_3 è il link register (R_1)

```
int inc (int x) {
    return (x+1);
}
        ↑ R2
```

```
int y = 123;
y = a+b;
i=0;
z = inc(y);
if(z=0) { i++ };
```

2048	inc :	ADD R_1, #1, R_2	// x, ret
2049		GOTO R_3	

1024	CLEAR	R_i	// i=0;
1025	MOV	R_y, R_1	// inc(y) preparare il parametro attuale
1026	CALL	R_{inc}, R_3	
1027	MOV	R_2, R_z	// z = inc(y)
1028			

main

IC = 1026 IC ← R_{inc} , IC+1 → R_3

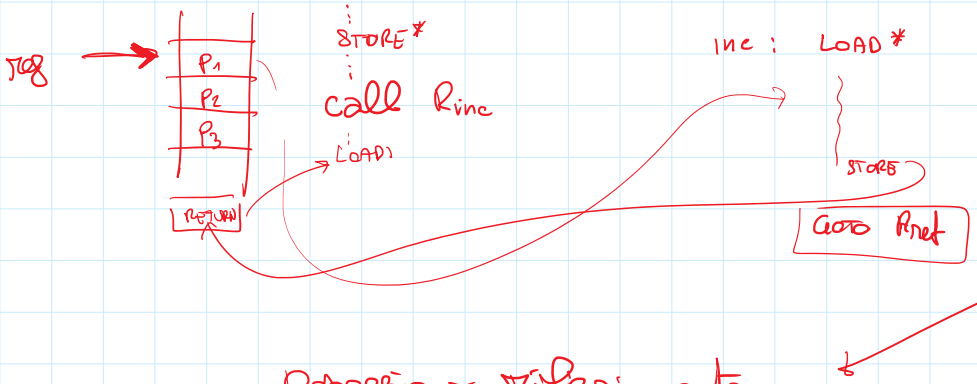
ADD IC = "inc" 2048
R3 = 1026+1 = 1027

```
ADD Ra, Rb, Ry
CLEAR Ri
MOV Ry, R1
CALL Rinc, R3
MOV R2, Rz
```

≡

```
ADD Ra, Rb, R1
CLEAR Ri
CALL Rinc, R3
MOV R2, Rz
```

passaggio dei param x memoria



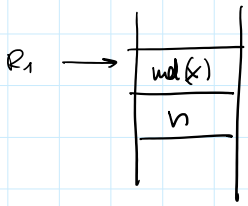

```
incvector(x[], n)
∀x[i] ← x[i]+1
```

passaggio x riferimento

```
incvector : MOV R0, Ri     CLEAR Ri
loop :      LOAD R1, Ri, R17
            INC R17
            STORE R1, Ri, R17
            INC Ri
            IF< Ri, R2, loop
cont :      GOTO Rret
```

```
void incvector(int *x, int n) {
    for(int i=0; i<n; i++)
        x[i] = x[i]+1;
    return;
}
```

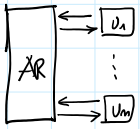
R3 link register

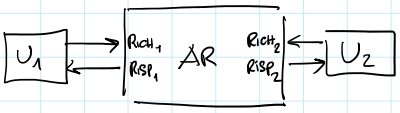



```
incvector : MOV R0, Ri
            LOAD R1, R0, Rbasex
            LOAD R1, #1, Rn
loop:       LOAD Rbasex, Ri, R17
            INC R17
            STORE Rbasex, Ri, R17
            INC Ri
            IF< Ri, Rn, loop
            GOTO Rret
```

arbitro a richieste multiple dedicate

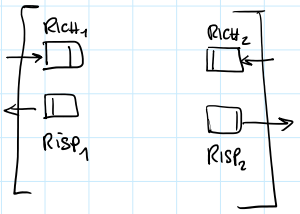
× 2 unità





Hp 1 U_i chiede

U_{AR} risponde ok



priorità a U_1

$\emptyset$. $(RICH_1, RICH_2 = \emptyset\emptyset)$ nop, $\emptyset$

$(=10)$ set $RISP_1$, reset $RICH_1$, 1

$(=01)$ set $RISP_2$, reset $RICH_2$, $\emptyset$

$(=11)$ set $RISP_1$, reset $RICH_1$, 1

PC 2 stati 1 bit x R 2 input

priorità ad U_2

1. $(RICH_1, RICH_2 = \infty)$, nop, 1

$(=10)$ set $RISP_1$, reset $RICH_1$, 1

$(=01)$ set $RISP_2$, reset $RICH_2$, $\emptyset$

$(=11)$ set $RISP_2$, reset $RICH_2$, $\emptyset$

T # ultimo unito servito

$0 \rightarrow U_1$

$1 \rightarrow U_2$

τ ?

$\tau = T_{WPO} + max\{T_{OPC}, T_{WPC} + T_{OPO}\} + \delta$

T_{WPO}: $\emptyset$; T_{OPC}: 2tp; T_{WPC}: $\emptyset$; T_{OPO}: 2tp

$= 2t_p + \delta = 3t_p$

$\emptyset$. $(RICH_1, RICH_2, T = \emptyset\emptyset-)$ nop, $\emptyset$

$(=10-)$ set $RISP_1$, reset $RICH_1$, $\emptyset \rightarrow T$, $\emptyset$

$(=01-)$ set $RISP_2$, reset $RICH_2$, $1 \rightarrow T$, $\emptyset$

$(=110)$ set $RISP_2$, reset $RICH_2$, $1 \rightarrow T$, $\emptyset$

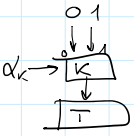
$(=111)$ set $RISP_1$, reset $RICH_1$, $\emptyset \rightarrow T$, $\emptyset$

τ ?

$T_{WPO} + max\{T_{OPC}, T_{WPC} + T_{OPO}\} + \delta$

T_{WPO}: $\emptyset$; T_{OPC}: $\emptyset$; T_{WPC}: t_K; T_{OPO}: t_p

$t_K = 2t_p$

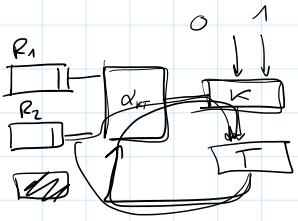


$5t_p$

α_{T_K}

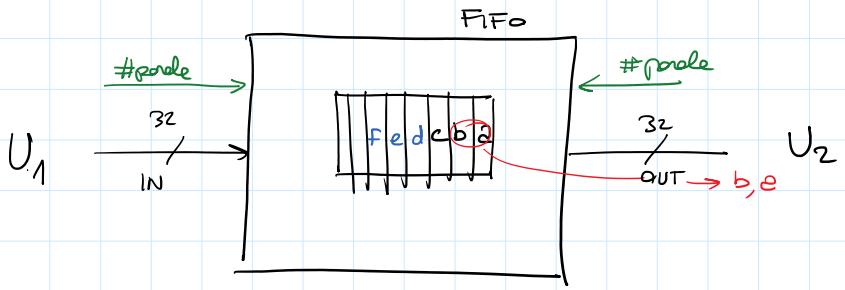
R_1 R_2 T	α_{T_K}	β_T
0 0 –	–	0
1 0 –	0	1
0 1 –	1	1
1 1 0	1	1
1 1 1	0	1

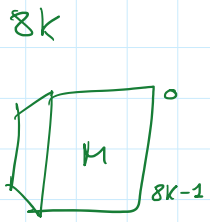
} $2t_p$



2013 - 2014 Materiale didattico

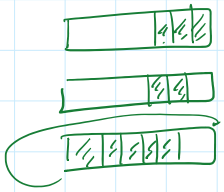
Homework 4 2)





Inserire a, b, c

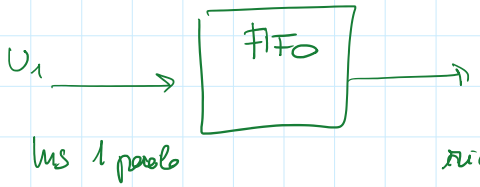
Ricevere x, y



Inserire d, e, f

Ricevere x, y, z, T

f, e, d, c



Ins 1 parola

Ricever 1 parola