AltriEsercizi4 (martedí 4/4/2017) Esercizio1

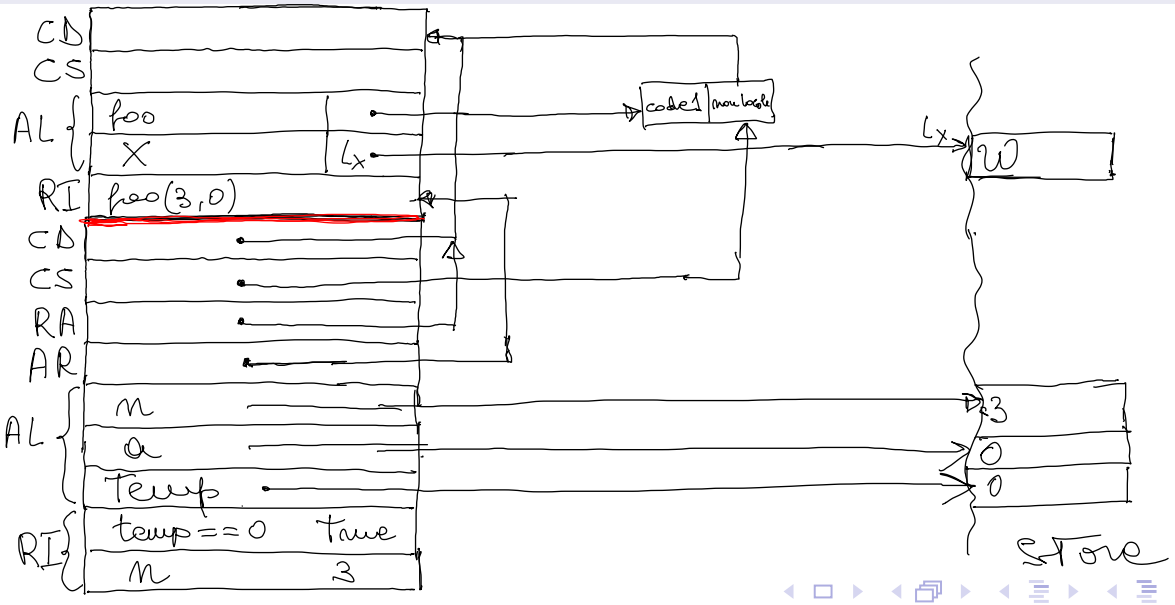
Esercizio (1)

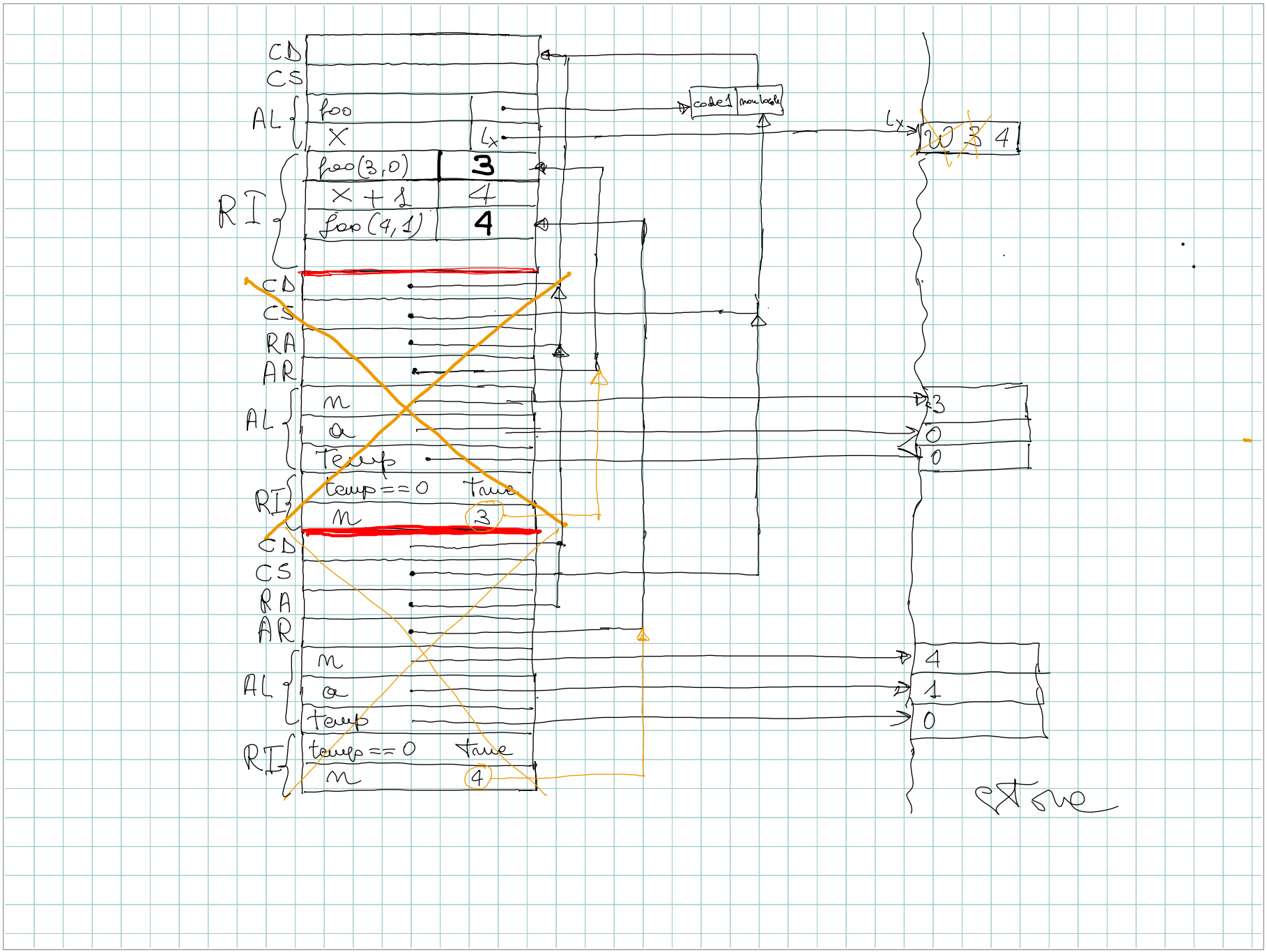
Si mostri la sequenza di AR generata dalla valutazione del seguente frammento di programma di un linguaggio a blocchi, scope statico, trasmissione per valore e, struttura C-like di comandi ed espressioni:

```
int foo(int n, int a){
   int temp = 0;
   if (temp == 0)return n;
   else return n + 1;
}
...
int x;
x = foo(3, 0);
x = foo(x + 1, 1);
```

code

In particolare si mostri come cambia il valore associato a `x` *nei vari AR.*




CD
CS
AL
foo
X
Lx
foo(3,0)
3
x + 1
4
foo(4,1)
4
RI
RA
AR
n
a
temp
temp == 0
true
code1
non locale
Lx
1 2 3 4
3
0
0
4
1
0
true

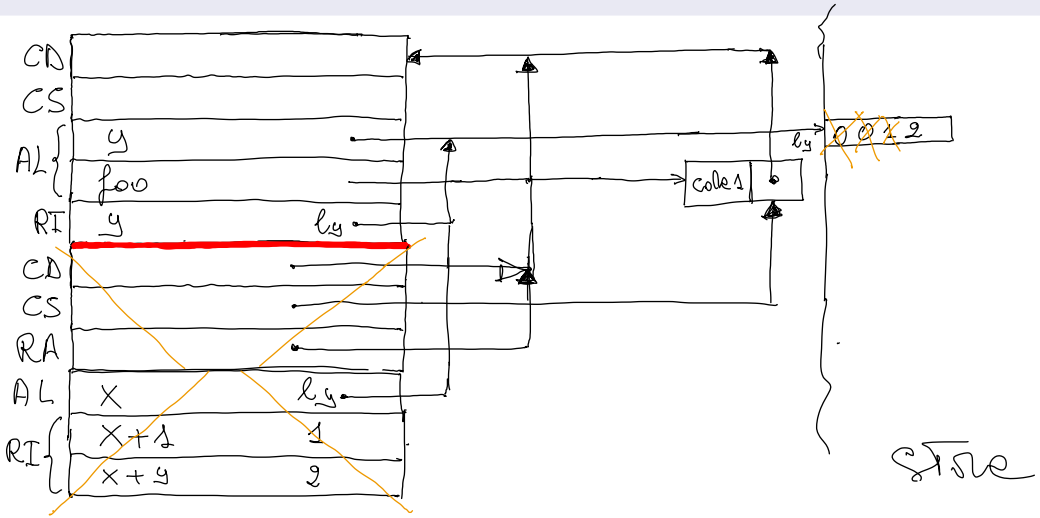
AltriEsercizi4 (lunedí 11/4/2016) Esercizio2

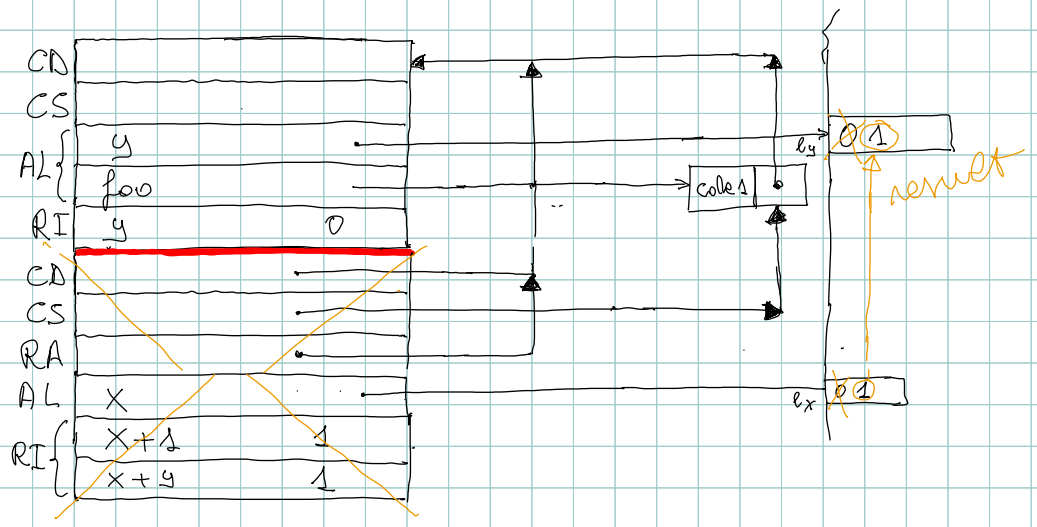
Esercizio (2)

Si mostri la sequenza di AR generata dalla valutazione del seguente frammento di programma di un linguaggio a blocchi, scope statico, trasmissione per xxxxx e, struttura C-like di comandi ed espressioni:

```
int y = 0;
void foo(int xxxxx x){
  x = x + 1;
  x = x + y;
}
y = 0;
foo(y);
```

In particolare si mostri come cambia il valore associato a `y` *nei vari AR, quando xxxxx è trasmissione per:*
(1) reference.
(2) value-result




CD
CS
AL
RI
y
foo
y
0
CD
CS
RA
AL
X
RI
X+1
X+y
1
1
code 1
l_y
l_x
0 1
0 1
result

AltriEsercizi4 (lunedí 11/4/2016) Esercizio3

Esercizio

(a) Si mostri la struttura degli AR generati dalla valutazione del seguente frammento di programma C:

```
int y = 0;
void foo(int * x){
  *x = *x + 1;
  *x = *x + y;
}
y = 0;
foo(&y);
```

(b) Si confronti tale struttura con quella ottenuta in esercizio 2.1.

AltriEsercizi4 (lunedí 11/4/2016) Esercizio4

Esercizio

Si utilizzi la copy rule *per mostrare come dovrebbe essere rifrasato il seguente frammento di programma di un linguaggio a blocchi, scope statico, trasmissione per valore e per nome e, struttura C-like di comandi ed espressioni:*

```
{...
    int summ(name int exp; name int i; value int start; value int stop){
      int acc = 0;
      for (i = start, i <= stop, i++) acc = acc + exp;
      return acc;
    }
  ...
      {int x = ...
       ...
       int y = summ((x − 1) ∗ (x − 1), x, 1, 10);
       ...
      }
      {int z = ...
       ...
       int y = summ(z ∗ z, z, 3, 12);
       ...
      }
```

① trasformiamo l'invocazione di una call invocazione

`.... = summ(1,10);`

dove summ è la funzione così riflessa:

```
int summ(value int start, value int stop){
    int acc = 0;
    for (i = start; i <= stop, i++) acc = acc + (x-1) * (x-1);
    return acc
}
```

la successiva invocazione di sum richiede una nuova e differente funzione sum2 ancora ottenuta per copy rule per rimpiazzamento sintattico del parametro formale trasmesso per nome

ESERCIZIO. Mostrare la struttura degli AR ottenuta utilizzando la trasmissione per nome (ovvero senza uso della copy rule) lasciando incompleti binding/valori dove il testo è ambiguo.