PROGETTO DI INFORMATICA

MODULO DI CALCOLATORI ELETTRONICI

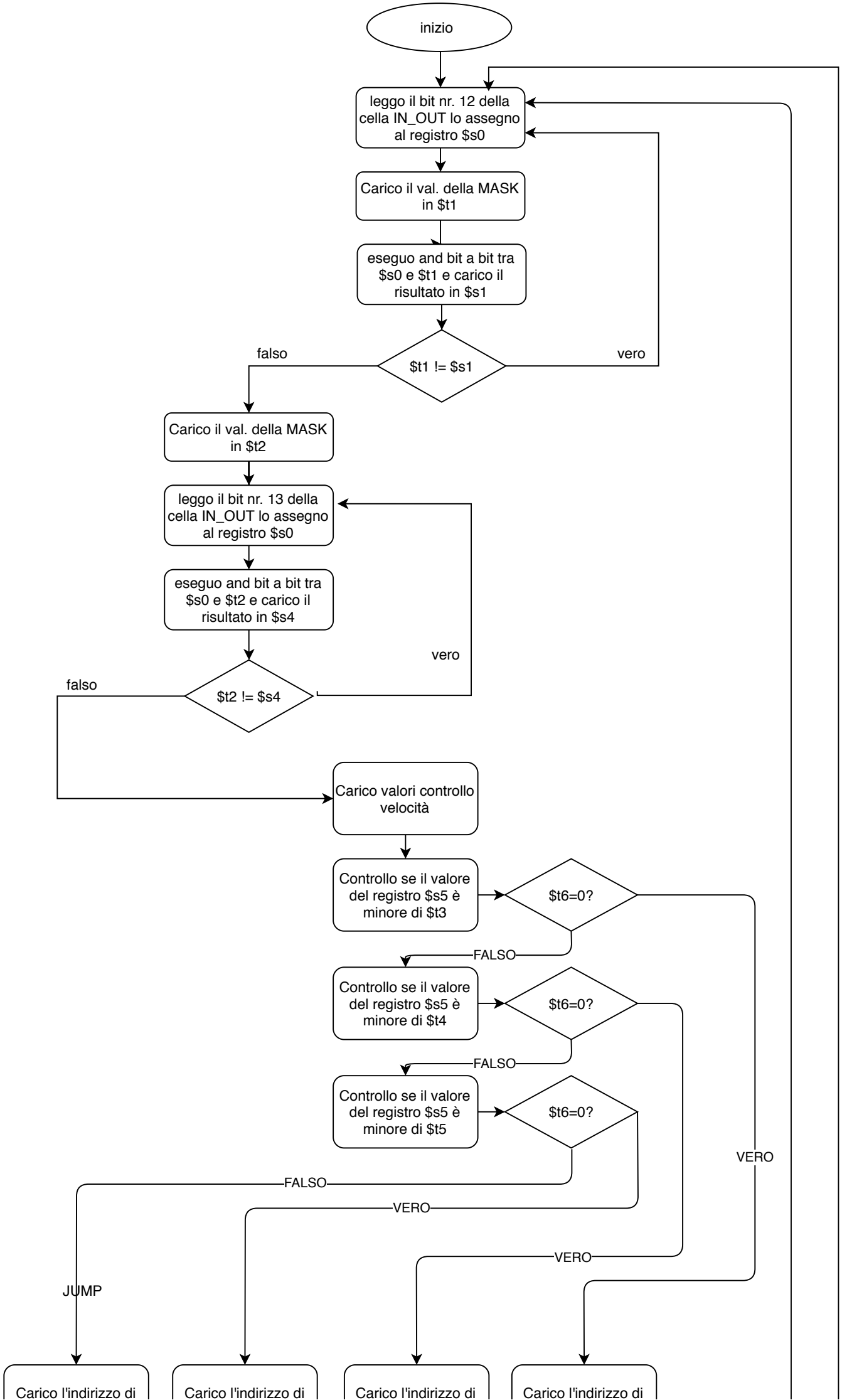
Università degli Studi di Bergamo a.a. 2019/2020

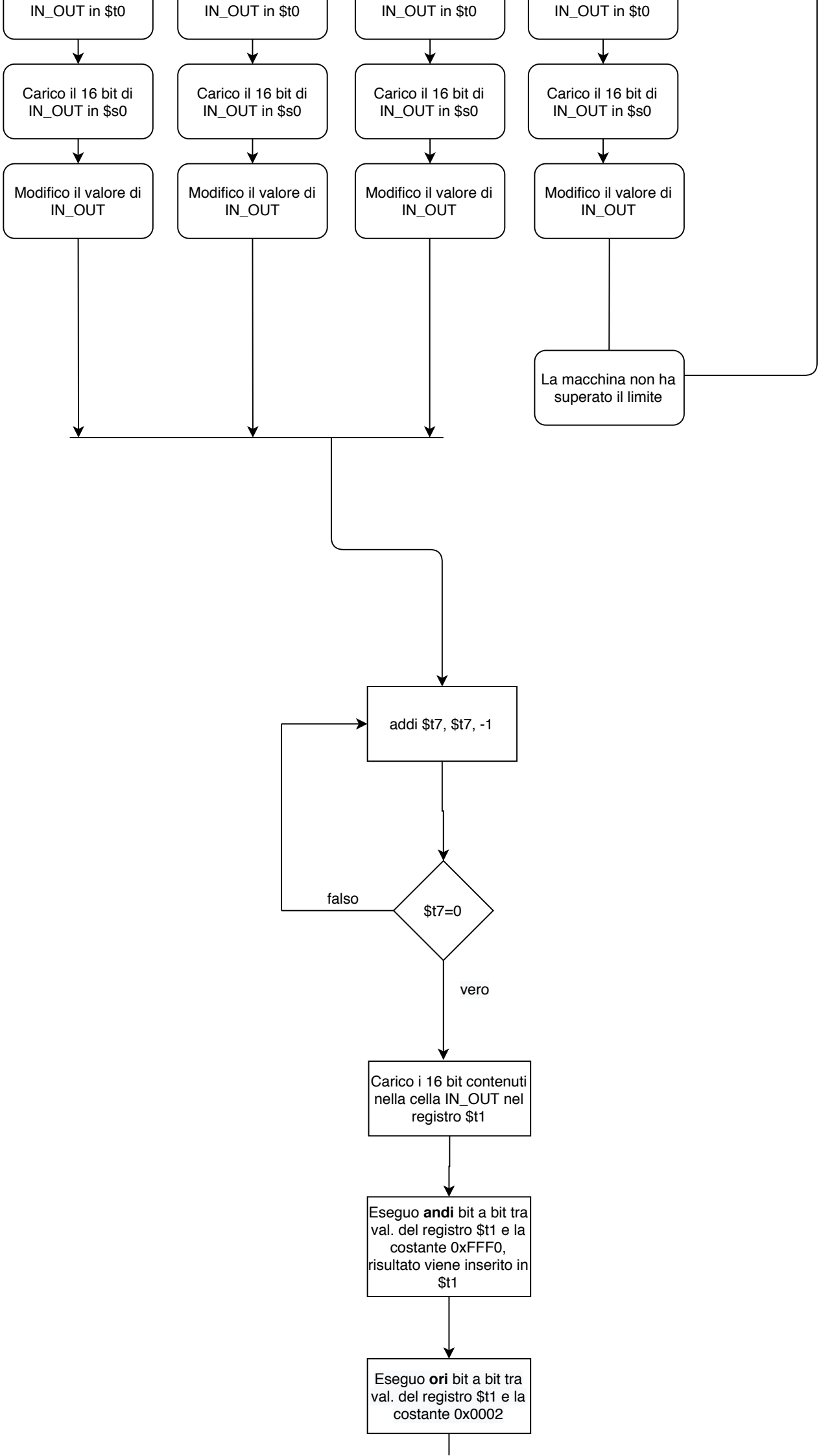
DOCENTE DEL CORSO:

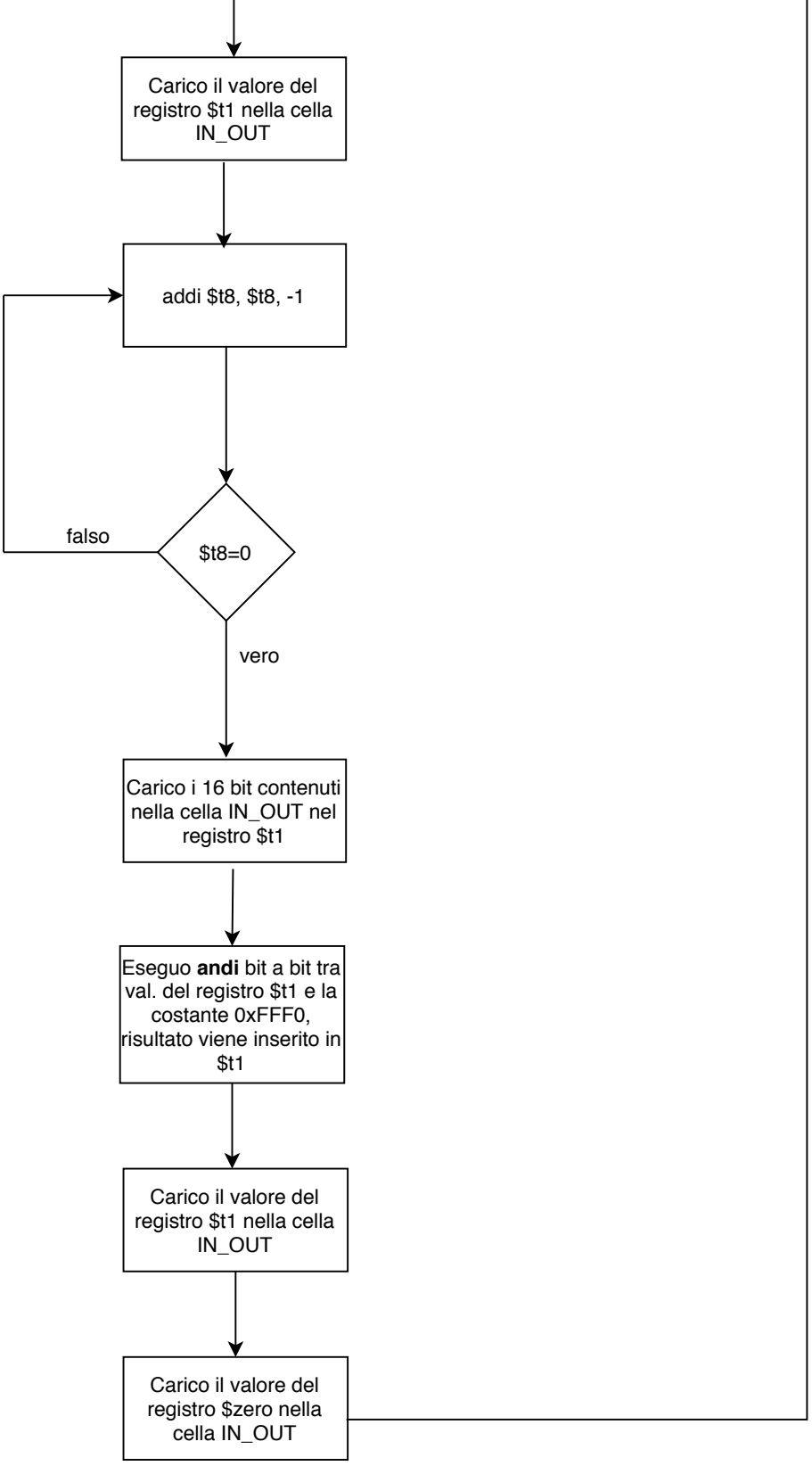
Ing. **Giuseppe Coldani**

COMPONENTI DEL GRUPPO:

Kenna Miriam Fatima 1058218 | **Dandis Iana** 1065350 | **Lorenzi Luca** 1068520


inizio
leggo il bit nr. 12 della cella IN_OUT lo assegno al registro $s0
Carico il val. della MASK in $t1
eseguo and bit a bit tra $s0 e $t1 e carico il risultato in $s1
$t1 != $s1
falso
vero
Carico il val. della MASK in $t2
leggo il bit nr. 13 della cella IN_OUT lo assegno al registro $s0
eseguo and bit a bit tra $s0 e $t2 e carico il risultato in $s4
$t2 != $s4
falso
vero
Carico valori controllo velocità
Controllo se il valore del registro $s5 è minore di $t3
$t6=0?
FALSO
Controllo se il valore del registro $s5 è minore di $t4
$t6=0?
FALSO
Controllo se il valore del registro $s5 è minore di $t5
$t6=0?
VERO
FALSO
VERO
VERO
JUMP
Carico l'indirizzo di
Carico l'indirizzo di
Carico l'indirizzo di
Carico l'indirizzo di


IN_OUT in $t0
IN_OUT in $t0
IN_OUT in $t0
IN_OUT in $t0
Carico il 16 bit di IN_OUT in $s0
Carico il 16 bit di IN_OUT in $s0
Carico il 16 bit di IN_OUT in $s0
Carico il 16 bit di IN_OUT in $s0
Modifico il valore di IN_OUT
Modifico il valore di IN_OUT
Modifico il valore di IN_OUT
Modifico il valore di IN_OUT
La macchina non ha superato il limite
addi $t7, $t7, -1
falso
$t7=0
vero
Carico i 16 bit contenuti nella cella IN_OUT nel registro $t1
Eseguo **andi** bit a bit tra val. del registro $t1 e la costante 0xFFF0, risultato viene inserito in $t1
Eseguo **ori** bit a bit tra val. del registro $t1 e la costante 0x0002


Carico il valore del registro $t1 nella cella IN_OUT
addi $t8, $t8, -1
$t8=0
falso
vero
Carico i 16 bit contenuti nella cella IN_OUT nel registro $t1
Eseguo **andi** bit a bit tra val. del registro $t1 e la costante 0xFFF0, risultato viene inserito in $t1
Carico il valore del registro $t1 nella cella IN_OUT
Carico il valore del registro $zero nella cella IN_OUT

#AUTOVELOX CODICE ESEGUIBILE

```
# passaggio dei dati alla memoria

           .data 0x10000000
IN_OUT:    .half 0x0000
           .text
```

 # AUTOVELOX

```
add $s5, $zero, $zero              #Contatore che viene utilizzato nei confronti
                                   #delle varie velocità
li $t7, 125000000                  #Per l'attesa di mezzo secondo
li $t8,  25000000                  #Per l'attesa di 100ms secondo

#-------------------------------------------------------------------------------
```

PRIMO SENSORE: è in attesa del passaggio dell'auto, il bit numero 12 va forzato a 1 per indicare il passaggio dell'auto

```
ciclo1: la $t0, IN_OUT                 #inserisco nel registro $st0 l'indirizzo
                                       #della cella IN_OUT
      li $t1, 0x1000                   #carico il valore di MASK in $t1
test1: lh $s0, 0($t0)                  #prelevo l'info da IN_OUT e la carico in s0
      and $s1, $s0, $t1                #and bit a bit tra la maschera e i bit di
                                       #IN_OUT, carico il risultato in $s1
      bne $s1, $t1, test1              #ritorna alla test se i due valori sono
                                       #diversi
#-------------------------------------------------------------------------------
```

SECONDO SENSORE: riceve il segnale del primo poi avvia il contatore finchè viene forzato ad 1 la linea 13 (uscita della macchina) per l'eventuale controllo e poi confronto della velocità

```
ciclo2:   li $t2, 0x2000               #carico il valore di MASK nel registro $t2
test2:    lh $s0, 0($t0)               #prelevo l'info da IN_OUT e la carico in s0
         and $s4, $s0, $t2             #and bit a bit tra la maschera e i bit di
                                       #IN_OUT carico il risultato in $s4
          addi $s5, $s5, 1
          bne $s4, $t2,test2           #se l'auto è uscita va al confronto
```

#CONTROLLO VELOCITA'

```
#Carico valori interi nei registri:

li $t3, 0x614658              #carico in $t3 il valore per 70km/h
li $t4, 0x55D4A8              #carico in $t4 il valore per 80km/h
li $t5, 0x4C4B40              #carico in $t5 il valore per 90km/h

#-------------------------------------------------------------------------------
#Confronto val ciclo con i registri temporanei:

slt $t6, $s5, $t3              #se $s5 ossia il registro contatore della prima
                               #parte è minore di $t3 allora $t6 è 1 sennò 0
beq $t6, $zero, Confronto0     #se $t6 è 1 vado al confronto finale senno proseguo


slt $t6, $s5, $t4              #se $s5 ossia il registro contatore della prima
                               #parte è minore di $t4 allora $t6 è 1 sennò 0
beq $t6, $zero, Confronto1     #se $t6 è 1 vado al confronto finale sennò proseguo


slt $t6, $s5, $t5              #se $s5 ossia il registro contatore della prima
                               #parte è minore di $t5 allora $t6 è 1 senn
beq $t6, $zero, Confronto2     #se $t6 è 1 vado al confronto finale sennò proseguo

j Confronto3                   #questo jump si raggiunge solo se la macchina ha
                               #superato i 90km/h

#-------------------------------------------------------------------------------
#Adesso per i confronti devo rendere i bit 9 e 8 di IN_OUT 00, 01,10,11 pertanto
uso 4 diversi valori per caricare il tutto in IN_OUT--> uso la notazione
esadecimale quindi: | 00 = 0xFCFF) | 01 = 0xFDFF) | 10 = 0xFEFF | 11 = 0xFFFF |

Confronto0:la $t0, IN_OUT        #carico l'indirizzo di IN_OUT in $t0
           li $s0, 0xFCFF        #carico i bit di $t6 in $s0
           sh $s0, 0($t0)        #carico i bit di nuovo in IN_OUT

           j ciclo1              #ritorno all'inizio perchè la macchina andava
                                 #a meno di 70km/h

Confronto1:la $t0, IN_OUT        #carico l'indirizzo di IN_OUT in $t0
           li $s0, 0xFDFF        #carico i bit di $t7 in $s0
           sh $s0, 0($t0)        #carico i bit di nuovo in IN_OUT

           j attesafoto          #passo ad attesafoto perchè ha sforato il
                                 #limite di 70km/h

Confronto2:la $t0, IN_OUT        #carico l'indirizzo di IN_OUT in $t0
           li $s0, 0xFEFF        #carico i bit di $t6 in $s0
           sh $s0, 0($t0)        #carico i bit di nuovo in IN_OUT

           j attesafoto          #passo ad attesafoto perchè ha sforato il
                                 #limite di 80km/h

Confronto3:la $t0, IN_OUT        #carico l'indirizzo di IN_OUT in $t0
           li $s0, 0xFFFF        #carico i bit di $t6 in $s0
           sh $s0, 0($t0)        #carico i bit di nuovo in IN_OUT

           j attesafoto          #passo ad attesafoto perchè ha sforato il
                                 #limite di 90km/h
```

#FOTOCAMERA: Dopo che la macchina è passata, attendiamo 0,5s e poi scattiamo la foto il cui impulso dura 100ms

```
attesafoto: addi $t7,$t7,-1          #Decremento di 1 per far passare mezzo secondo
beq $t7,$zero,camera1                #Se il mezzo secondo è passato,salto acamera1
                                     #per scattare, altrimenti aspetto ancora

j attesafoto

camera1:    lh $t1, 0($t0)             #Carico i 16 bit di IN_OUT per poter
                                       #lavorare su quelli necessari
            andi $t1,$t1,0xFFF0        #Modifico valore del bit 1 per
            ori $t1,$t1,0x0002         #comandare lo scatto della fotocamera
            sh $t1,0($t0)              #Scrivo in Memory($t0+0)il valore di $t1

            j impulso                  # Salto per l'attesa di 100ms (impulso)

impulso:    addi $t8,$t8,-1            #Decremento di 1 per far passare 100ms
            beq $t8,$zero,camera2      #Se i 100 ms sono passati, salto a camera2
                                       #per terminare lo scatto

            j attesafoto               #Altrimenti aspetto ancora

camera2:    lh $t1, 0($t0)             #Carico i 16 bit di IN_OUT
            andi $t1, $t1, 0xFFF0      #Termino lo scatto della fotocamera
            sh $t1, 0($t0)             #Scrivo in Memory($t0+0)il valore di $t1
            sh $zero, 0($t0)           #Riporto alla condizione iniziale IN_OUT
                                       #prima di tornare a controllare il primo
                                       #sensore
            j ciclo1                   #Torno a ciclo1: aspetto una nuova macchina
```

ALCUNI CHIARIMENTI

1. REALIZZAZIONE DELL'ATTESA

Per l'attesa di mezzo secondo:
carico nel registro **$t7** tramite l'istruzione **li** il valore: **125000000**
sapendo appunto che lavoro su un'architettura con **il clock pari a 500 MHz**
ottengo il valore della variabile da inserire nel registro dalla formula:

$$val = 0{,}5 / (2 * 5x10^8\ Hz) = 125000000$$

Nella quale ho supposto per semplicità che il tempo delle tre istruzioni **li, addi, bni** sia lo stesso.
Per l'attesa di 100ms: il ragionamento è simile sostituendo i valori

$$val = 0{,}1 / (2 * 5x10^8\ Hz) = 25000000$$

2. IL PRINCIPIO DI FUNZIONAMENTO DEI DUE SENSORI:

Utilizzo della tecnica della maschera:
l'idea è quella di caricare nella MASK il valore che serve a noi per poi fare un and bit a bit tra la maschera e i bit di IN_OUT, visto che l'operazione logica and da in uscita 1 solo nel caso in cui entrambi i bit valgono 1, cosi facciamo il programma ritornare al ciclo 1 per ricontrollare il sensore 1 ogni volta che dal confronto tra i bit 12 della cella IN_OUT e MASK ne risulta che sono diversi.

1°SENSORE:
Dal testo del problema sappiamo che il bit numero 12 sta ad indicare il rilevamento del passaggio della macchina.

- quindi la linea 12 vale della cella a 16 bit IN_OUT:
  = 0 – se la macchina non è ancora passata
  = 1 – se la macchina è passata

IN_OUT

0	0	0	1/0	0	0	0	0	0	0	0	0	0	0	0	0

MASK

0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Quindi il valore della MASK in esadecimale da inserire nel registro è: 1000

2° SENSORE:
Per il secondo sensore il ragionamento è simile con la differenza che questa volta prendo in considerazione il bit numero 13

IN_OUT

0	0	1/0	0	0	0	0	0	0	0	0	0	0	0	0	0

MASK

0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Quindi il valore della MASK in esadecimale da inserire nel registro è: 2000

3. SPIEGAZIONE CONTROLLO VELOCITA'

1° BLOCCO

Nel primo blocco di istruzioni carico semplicemente i valori delle operazioni al secondo che il processore può fare in **3 registri.**
Il mio ragionamento è stato che se il processore ha una velocità di **500Mhz** è in grado di eseguire 500milioni di operazioni al secondo.
Siccome il **ciclo 2** del secondo sensore contiene **4 operazioni** può essere eseguito **125'000'000** volte al secondo la distanza tra i sensori è di 1 metro quindi se la macchina va a **70km/h** ci mette **0,051s** a percorrere la distanza, se va a **80km/h** ci mette **0,045s** e se va a 90km/h ci mette **0,04s** quindi:
-se la macchina andasse a **70km/h** si potrebbero eseguire solo **6'375'000** (in Esadecimale **614658**) cicli 2;
-se la macchina andasse a **80km/h** si potrebbero eseguire solo **5'625'000**(in Esadecimale **55D4A8**) cicli 2;
-se la macchina andasse a **90km/h** si potrebbero eseguire solo **5'000'000** (in Esadecimale **4C4B40**) cicli 2;
siccome i registri temporanei utilizzati nella prima parte arrivano a **$t2** sono partito da **$t3.**

2° BLOCCO

Nel secondo blocco di codice vado a confrontare il valore del registro $s5 usato dalla collega Yana come contatore per contare il numero di volte che il ciclo viene eseguito.
Le operazioni usate sono un semplice **slt** che verifica se il valore del ciclo è inferiore del valore da me imposto ai registri temporanei e in caso negativo il registro temporaneo **$t6** assume 0 come valore, di conseguenza tramite un confronto reso possibile dall'operatore **beq** si può saltare all'ultimo blocco di istruzioni.

3° BLOCCO

Nell'ultimo blocco di istruzioni rendo i bit 9 e 8 di IN_OUT 00, 01,10,11 in base alla velocità dell'auto, pertanto uso 4 diversi valori per caricare il tutto in **IN_OUT** -> per **00=0xFCFF;** per **01=0xFDFF;** per **10=0xFEFF;** per **11=0xFFFF.**
Questi numeri però vanno a cambiare i valori di tutti i bit del registro **IN_OUT** settandoli ad 1(tranne ovviamente l'8 e il 9 che variano in base alla velocità).
Questa scelta è dovuta al fatto che per motivi di chiarezza espositiva del progetto la collega Miriam ha deciso che per rendere più evidente il settaggio del bit 1 ad valore 0 o 1 fosse meglio cambiare i valori di tutti gli altri bit ad 1 ed è questa la ragione di questo passaggio.
Per richiamare **IN_OUT** utilizzo lo stesso metodo usato nella parte dei sensori e uso gli stessi registri in modo tale da non creare confusione.

Quindi carico l'indirizzo di **IN_OUT** in **$t0** e poi i 16 bit nel registro **$s0** in modo tale da poterli modificare con i valori da me calcolati in modo tale da settare tutti i bit a 1 tranne l'8 e il 9 che variano in base alla velocità.

4. **IL PRINCIPIO DI FUNZIONAMENTO DELLA TELECAMERA** si basa su:

1. Conoscenza che la macchina è passata, ed a quale velocità (indicazioni contenute nella half-word nel registro $t0
2. Attesa 500ms
3. Scatto della foto con impulso pari a 100ms
4. Terminazione del processo: riporto tutto alla condizione iniziale

Inizialmente utilizzo il valore caricato inizialmente in **$t7** per eseguire un'azione di decremento di 1. Questa operazione viene ripetuta (tramite una **jump**) per n volte tali da consentire un'attesa di 500ms. Una volta che il registro $t7 ha valore pari a 0, tramite una **beq** salto all'istruzione avente etichetta **camera1.**

Successivamente carico i 16 bit di **IN_OUT** in modo da poter lavorare su quelli a me necessari (in particolare bit1). Utilizzo una combinazione di due istruzioni, **andi** e ori (and e or tramite uso di una costante e non entrambi registri).
Ecco l'implementazione:
uso come esempio il caso in cui abbiamo come convenzione di velocità 01

In **$t1** avrò il valore esadecimale: **0xFDFF**

1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1

andi $t1 , $t1, 0xFFF0

1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0

Quindi otteniamo:

1	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0

Così facendo otteniamo i primi 4 bit che sono pari a 0. Fotocamera pronta per scattare. Eseguo **ori $t1,$t1,0x0002**

1	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Quindi otteniamo:

1	1	1	1	1	1	0	1	1	1	1	1	0	0	1	0

Così facendo abbiamo settato il bit1 a 1. Proprio quello che ci interessava.

Il valore ottenuto nel registro **$t1**, lo scriviamo nella cella avente indirizzo $t0 più offset 0. Possiamo quindi saltare all'istruzione avente etichetta impulso.

Inizialmente utilizzo il valore caricato inizialmente in $t8 per eseguire un'azione di decremento di 1. Questa operazione viene ripetuta (tramite una **jump**) per n volte tali da consentire un'attesa di **100ms**. Una volta che il registro **$t8** ha valore pari a 0, tramite una **beq** salto all'istruzione avente etichetta camera2.

Successivamente carico i 16 bit di **IN_OUT** in modo da poter lavorare su quelli a me necessari (in particolare bit1). Utilizzo un'istruzione **andi** per forzare il bit alla condizione di normalità.

Ecco l'implementazione:

In **$t1** avrò il valore esadecimale: **0xFDF2**

1	1	1	1	1	1	0	1	1	1	1	1	0	0	1	0

andi $t1, $t1, 0xFFF0

1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0

Quindi otteniamo:

1	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0

Così facendo otteniamo i primi 4 bit che sono pari a 0. Fotocamera torna alla situazione dipartenza, ed è pronta per scattare.