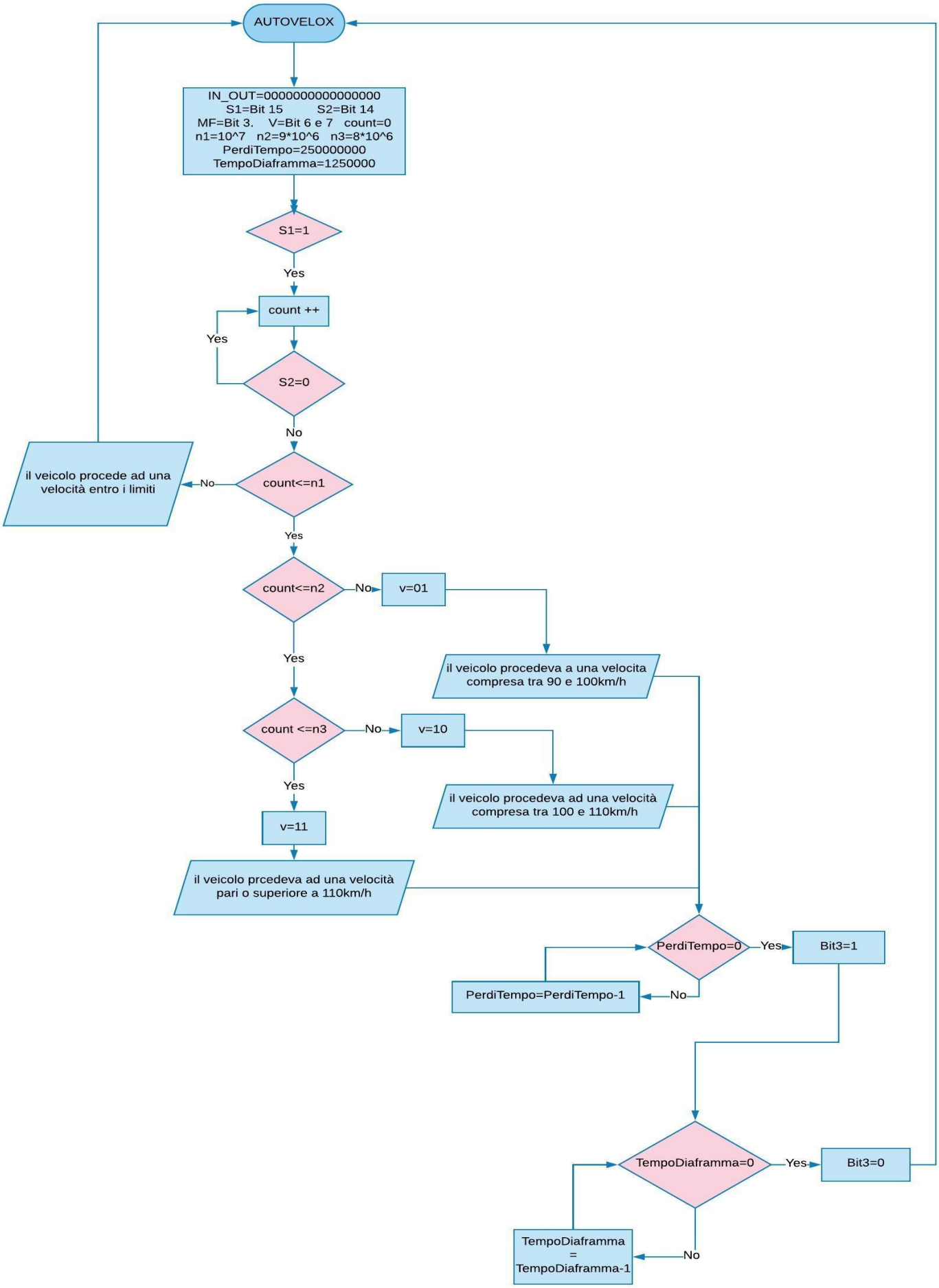
RELAZIONE AUTOVELOX

Aurora Zanenga & Laura Gozzi

Anno Accademico 2017-2018 – Progetto Informatica 1 – mod. Calcolatori Elettronici per Prova Orale

Esercizio 004–17–18

FLOW CHART:


AUTOVELOX
IN_OUT=0000000000000000
S1=Bit 15 S2=Bit 14
MF=Bit 3. V=Bit 6 e 7 count=0
n1=10^7 n2=9*10^6 n3=8*10^6
PerdiTempo=250000000
TempoDiaframma=1250000
S1=1
Yes
count ++
Yes
S2=0
No
il veicolo procede ad una velocità entro i limiti
No
count<=n1
Yes
count<=n2
No
v=01
il veicolo procedeva a una velocita compresa tra 90 e 100km/h
Yes
count <=n3
No
v=10
il veicolo procedeva ad una velocità compresa tra 100 e 110km/h
Yes
v=11
il veicolo prcedeva ad una velocità pari o superiore a 110km/h
PerdiTempo=0
Yes
Bit3=1
PerdiTempo=PerdiTempo-1
No
TempoDiaframma=0
Yes
Bit3=0
TempoDiaframma = TempoDiaframma-1
No

LEGENDA VARIABILI:

IN_OUT : una cella a 16 bit, contenente a sua volta i bit che ci interessano.
S1: primo sensore del dispositivo, letto dal bit 15 della cella.
S2: secondo sensore del dispositivo, letto dal bit 14 della cella.
MF: macchina fotografica, comandata dal bit 3 della cella.
V: velocità, rappresentata dai bit 6 e 7 della cella, in base al valore che questi ultimi assumono, si può dedurre la gravità dell'infrazione.
Count: variabile contatore che si incrementa ogni 4ns (tempo di clock) dal momento di attivazione del primo sensore fino al momento in cui si attiva il secondo.
N1: costante che rappresenta il numero di volte che count si incrementerebbe se un'ipotetica auto procedesse ad esattamente 90km/h (1m*250MHz/25m/s)
N2: costante che rappresenta il numero di volte che count si incrementerebbe se un'ipotetica auto procedesse ad esattamente 100km/h
N3: costante che rappresenta il numero di volte che count si incrementerebbe se un'ipotetica auto procedesse ad esattamente 110km/h
PerdiTempo: variabile che serve per far scattare la foto dopo 1s dal passaggio dell'auto dal secondo sensore, è inizializzata a 250000000 perché questo è il numero di volte che deve essere decrementata affinché sia passato 1s (1s/4ns=250000000).
TempoDiaframma: variabile che serve per tenere aperto l'otturatore della macchina fotografica per 500ms. Il suo valore è fissato a 1250000 perché questo è il numero di volte che deve essere decrementata affinché siano passati 500ms da quando la macchina fotografica è stata accesa.

CALCOLO VALORI VARIABILI:

$$n1 = \frac{1m \times 250MHz}{25m/s} = 10^7$$

$$n2 = \frac{1m \times 250MHz}{27,\bar{7}m/s} = 9 \times 10^6$$

$$n2 = \frac{1m \times 250MHz}{30,\bar{5}m/s} = 8 \times 10^6$$

$$PerdiTempo = \frac{1s}{4ns} = 250000000$$

$$TempoDiaframma = \frac{500ms}{4ns} = 125000000$$

PROCEDIMENTO:

- L'auto passa dal primo sensore (S1) e il programma si attiva incrementando 'count' fino a quando l'auto passa dal secondo sensore.

- Si paragona il valore di 'count' a N1, se 'count' è maggiore di N1, allora l'auto sta viaggiando ad una velocità inferiore a 90 km/h (e si ritorna all'inizio del programma), se invece 'count' è minore di N1, allora la velocità dell'auto è maggiore di 90km/h e si procede con la verifica del range di infrazione (Ricordiamo che stiamo confrontando il numero di volte in cui viene incrementata la variabile durante il passaggio dell'auto, perciò se la variabile 'count' è più piccola di N1 significa che la macchina è passata più velocemente e che quindi andava ad una velocità maggiore).

- Successivamente si controlla la gravità dell'infrazione, valutando se l'auto:
  - Viaggia tra i 90 km/h e i 100 km/h, quindi se *N1 > count ≥ N2*, in questo caso si attribuisce a 'V' il valore 01.
  - Viaggia tra i 100 km/h e i 110 km/h, quindi se *N2 > count ≥ N3*, in questo caso si attribuisce a 'V' il valore 10.
  - Viaggia a velocità superiore a 110 km/h, quindi se *count < N3*, in questo caso si attribuisce a 'V' il valore 11.

- Nel caso in cui la velocità sia maggiore di 90 km/h si attiva la macchina fotografica dopo aver decrementato PerdiTempo da 25000000 a 0 (per fare ciò serve 1s).

- La macchina fotografica deve rimanere attiva per 500ms (impulso della macchina fotografica), perciò si decrementa TempoDiaframma da 125000000 a 0 (per fare ciò servono 500ms), una volta effettuata la foto questa verrà nuovamente disattivata.

- Una volta spenta la macchina fotografica, il programma riparte dall'inizio (torna al punto in cui IN_OUT=0000000000000000).

CODICE DEL PROGRAMMA:

Abbiamo scelto di inserire all'interno della relazione anche parte del codice in linguaggio ad alto livello (java) per una migliore comprensione del programma:

```
            .data 0x10010000
IN_OUT:     .half 0x0000                #carico 16 bit a 0 in memoria
            .text 0x00400000
            la $t0, IN_OUT              #carico in $t0 l'indirizzo di IN_OUT
            lh $s0, 0($t0)              #dentro $s0 ho tutti i 16 bit

cod_1:      la $t0, IN_OUT
            lh $s0, 0($t0)

            lui $s1, 0x05F5             #n1 (val di count se V=90km/h)
            ori $s1, $s1, 0xE100

            lui $s2, 0x055D             #n2 (val di count se V=100km/h)
            ori $s2, $s2, 0x4A80

            lui $s3, 0x04C4             #n3 (val di count se V=110km/h)
            ori $s3, $s3, 0xB400

            lui $s4, 0x0000             #count
            ori $s4, $s4, 0x0000

            lui $s5, 0x0013             #addi $s5, $zero, 1250000
            ori $s5, $s5, 0x12D0        #var per perdiTempo 500ms (tempo diaframma)

            lui $s6, 0x0EE6             #addi $s6, $zero, 250000000
            ori $s6, $s6, 0xB280        #var per perdiTempo 1s

            lui $t7, 0x0000
            ori $t7, $t7, 0x8000
            addi $t8, $zero, 0x4000     #per il confronto del 15esimo bit

loop1:      beq $s0, $t7, loop          #continua a controllare se sensore1=0x8000
            j loop1
```

```
loop:       addi $s4, $s4, 1            #count++
            beq $s0, $t8, check         #incrementa fintantoche sensore2=0x4000
            j loop

check:      slt $t3, $s1, $s4           #1 se v<90 - 0 se v>90
            beq $t3, 1, fine            #se $t3=1 goto vel1(v<90)
            beq $t3, $zero, vel         #se $t3=0 goto vel2(v>90)

fine:       j cod_1                     #ha terminato e aspetta la prossima macchina

vel:        slt $t5, $s2, $s4           #se v<100 --> $t5=1, se v>100 --> $t5=0
            beq $t5, 1, fine_1
            beq $t5, $zero, vel_1

vel_1:      slt $t5, $s3, $s4           #se v<110 --> $t5=1, se v>110 --> $t5=0
            beq $t5, 1, fine_2
            beq $t5, $zero, fine_3

fine_1:     ori $s0, $s0, 0x0080        #metto i bit 6 e 7 a 01
            j perdiT

fine_2:     ori $s0, $s0, 0x0040        #metto i bit 6 e 7 a 10
            j perdiT

fine_3:     ori $s0, $s0, 0x00C0        #metto i bit 6 e 7 a 11
            j perdiT

perdiT:     addi $s6, $s6, -1           #decrementa s6 fintantoche s6=0
            beq $s6, $zero, pht         #salta a pht quando arriva a 0
            j perdiT

pht:        ori $s0, $s0, 0x0008        #metto il bit 3 a 1 (MF)
            j perdiT1

perdiT1:    addi $s5, $s5, -1           #decrementa s5 fintantoche s5=0
            bnq $s5, $zero, perdiT1
            sh $t0, 4($s0)
            j cod_1
```

LEGENDA DEI REGISTRI UTILIZZATI NEL CODICE:

Registro	**Scopo**
$t0	Contiene l'indirizzo della mia half word
$s0	Contiene i 16 bit all'indirizzo contenuto in $t0, inizialmente tutti a 0
$t1	Valore del sensore 1
$t2	Valore del sensore 2
$s1	Costante n1 – valore che assumerebbe 'count' se un'auto passasse alla velocità di 90 km/h
$s2	Costante n2 – valore che assumerebbe 'count' se un'auto passasse alla velocità di 100 km/h
$s3	Costante n3 – valore che assumerebbe 'count' se un'auto passasse alla velocità di 110 km/h
$s4	Variabile 'count'
$s5	Costante che rappresenta il tempo diaframma
$s6	Costante da utilizzare nel ciclo perditempo
$t3	Contiene il valore della 'slt' nel controllo della velocità (maggiore o minore di 90)
$t4	Contiene la 'or' fatta nel momento dello scatto della macchina fotografica
$t5	Contiene il valore della 'slt' nel controllo della velocità tra 90 e 100, tra 100 e 110 e maggiore di 110
$t6	Contiene la 'or' che modifica i bit 6 e 7 per 90 < V 100, per 100 < V < 110 e per V > 110

SPIEGAZIONE PORZIONI DI CODICE:

Le prime istruzioni del codice (compreso il blocco etichettato *IN_OUT*) hanno lo scopo di inserire l'half word (i 16 bit su cui lavoreremo) all'interno della memoria.

Con la direttiva `.data` e con la direttiva `.half` specifico rispettivamente dove voglio salvare il mio dato e che il mio dato è una half word con i 16 bit tutti a 0. Carico infine i miei 16 bit a 0 nel registro `$s0`.

Nella sezione di codice etichettata *cod_1* si ha per prima cosa il caricamento della mia sequenza di bit, in secondo luogo avremo la dichiarazione di tutte le variabili e costanti di cui necessiterò durante l'esecuzione del programma:
- Le costanti `$s1`, `$s2`, `$s3` che rappresentano il numero di volte in cui 'count' si incrementerebbe se la velocità di un'ipotetica auto fosse rispettivamente di 90km/h, 100 km/h e 110 km/h.
Questi valori verranno utilizzati successivamente per effettuare i confronti e inquadrare la fascia della velocità a cui andava la macchina in questione.
Dato che i valori che inseriremo in questi registri superano il valore di 16 bit dovremo inserirli tramite le istruzione `lui` e `ori` che settano rispettivamente i 16 bit più significativi e i 16 bit meno significativi.
- La variabile 'count' (inizializzata a 0, anch'essa a 32 bit) che verrà incrementata successivamente all'interno del ciclo '*loop*' a partire da quando viene attivato il primo sensore e smette nel momento in cui viene attivato il secondo, sarà proprio il valore di questa variabile 'count' ad essere confrontata con i valori dei registri `$s1`, `$s2` e `$s3`.
- i registri `$s5` e `$s6` (sempre a 32 bit) che corrispondono alle variabili che verranno decrementate per i due cicli perdi-tempo, `$s5` per il tempo di diaframma, quindi il tempo necessario allo scatto della foto (500ms che equivalgono a 1250000 decrementi di 'count'), mentre `$s6` è il tempo che bisogna aspettare prima di avviare la macchina fotografica dopo lo spegnimento dei due sensori (equivale a 1s = 250000000).
- I registri `$t8` e `$t9` che rappresentano le costanti con cui confronteremo la nostra sequenza di bit per controllare se i due sensori sono stati attivati o meno: il registro `$t7` contiene il valore 0x8000 che equivale al 16-esimo bit della sequenza settato a 1, il registro `$s8` invece contiene il valore 0x4000 che equivale al 15-esimo bit della sequenza settato a 1.

Nella sezione di codice '*loop1*' viene effettuato continuamente un controllo sulla sequenza di bit per vedere se è stato attivato il primo sensore (e nel caso iniziare l'incremento di 'count').
Per fare questo dobbiamo confrontare il valore della nostra sequenza (contenuta in `$s0`) con il valore della costante `$t7` dichiarata in '*cod_1*', se sono uguali salta alla sezione '*loop*', sennò ripeti il controllo.

La sezione di codice '*loop*' è il ciclo che mi incrementa 'count' a partire da quando si attiva il primo sensore fino a quando non si disattiva (quindi quando l'auto è passata).
Si esce dal ciclo nel momento in cui il secondo sensore diventa uguale ad 1 (0x4000). Se questa uguaglianza è verificata salto alla porzione di codice etichettata come '*check*'.

Qui (all'interno di '*check*') verrà effettuato il controllo sulla velocità, nel caso in cui la velocità sia minore di 90 km/h (quindi il numero di volte in cui viene incrementato 'count' è maggiore (la macchina ha impiegato più tempo per attraversare lo spazio tra i due sensori) rispetto alla costante contenuta in `$s1`) significa che la macchina è andata ad una velocità adeguata per quel tratto di strada, in questo caso ritorno a '*cod_1*', quindi all'inizio del codice attendendo il passaggio dell'auto successiva senza effettuare modifiche sulla sequenza di bit.
Anzi che inserire direttamente nella '`beq`' della sezione '*check*' il salto incondizionato a '*cod_1*' si è scelto di inserire un'ulteriore sezione di codice denominata '*fine*' per motivi di simmetria tra le 4 possibili casistiche verificabili in base alla velocità dell'auto.
Nel caso in cui la velocità sia maggiore di 90 km/h si viene reindirizzati ad un ulteriore controllo sulla velocità che verrà effettuato nella sezione '*vel*'.
Quando arrivo qui so già che la mia velocità è maggiore di 90 km/h e di conseguenza mi resta da effettuare un controllo sulla fascia a cui appartiene la velocità dell'auto in questione e quindi determinare la gravità dell'infrazione.
Effettuo il primo controllo:
Se la mia velocità è minore di 100 km/h significa che sono nella fascia *90 km/h < V < 100 km/h*, in questo caso salto alla sezione '*fine*_1'.

Nel caso in cui la mia velocità fosse maggiore di 100 km/h mi troverei a dover effettuare un'ulteriore controllo, in tal caso quindi verrei reindirizzata alla sezione '*vel*_1'

Nella sezione '*vel*_1' effettuo l'ennesimo controllo sulla mia velocità, so già che è maggiore di 100 km/h quindi devo controllare se è maggiore o minore di 110 km/h, effettuo il controllo con la costante `$s2` (che rappresenta il valore che assumerebbe 'count' se un'ipotetica auto passasse a 110 km/h), nel caso in cui count > `($s2)` (quindi *V < 110 km/h*) verrei reindirizzata alla sezione 'fine_2', mentre nel caso in cui 'count' fosse minore del contenuto di `$s2`, quindi la velocità fosse maggiore di 110 km/h, allora in tal caso verrei reindirizzata alla sezione '*fine*_3', ultima possibile casistica.

Se arrivo alla sezione '*fine*_1' significa che la mia velocità è compresa tra 90 km/h e 100 km/h, il comportamento previsto per questa casistica consiste nel sostituire i bit numero 6 e 7 con i rispettivi valori 0 e 1.
Per fare ciò effettuo una '`ori`' tra il valore della mia sequenza di bit originale (con il bit 3 a 1 a causa dello scatto della macchina fotografica) e una costante appositamente scelta per cambiare questi due bit (0x0080 = 0000 000 **10**00 0000), quindi l'operazione eseguita è:
0000 0000 0000 1000 | 0000 000 **10**00 0000 = 0000 0000 1000 1000 = 0x0088
Al termine di questa '`ori`' ho la sequenza di bit corretta per la casistica *90 km/h < V < 100 km/h*
Terminata la modifica della sequenza devo scattare la foto, salto quindi al ciclo perditempo (*perdiT*).

Per quanto riguarda la sezione '*fine*_2' assumeremo il medesimo comportamento utilizzato nella sezione '*fine*_1',modificando però i bit 6 e 7, anzi che in 0 e 1 in 1 e 0, anche per questo caso abbiamo definito una costante che mi vada a cambiare solo i bit interessati tramite la `ori`, l'operazione è la seguente:
0000 0000 0000 1000 | 0000 0000 **01**00 0000 = 0000 0000 0100 1000 = 0x0048
Come nella casistica precedente al termine della '`ori`' è stata messa una '`jump`' che rimanda al ciclo '*perdiT*'.

La sezione '*fine*_3' ha un comportamento equivalete, i bit 6 e 7 verranno sostituiti dai bit 1 e 1 tramite una '`ori`' effettuando la seguente operazione:
0000 0000 0000 1000 | 0000 0000 **11**00 0000 = 0000 0000 1100 1000 = 0x00C8
Terminata la modifica sella sequenza saremo nuovamente reindirizzati al ciclo '*perdiT*'.

La sezione '*perdiT*' (perdi tempo) ha lo scopo di far passare un secondo prima di attivare la macchina fotografica, questo ciclo è stato concepito per decrementare la costante contenuta in `$s6` (dichiarata a inizio programma nella sezione *cod_1*) fino ad arrivare a 0, il valore della costante è stato scelto appositamente perché impiegasse un secondo a decrementarsi fino ad arrivare a 0.
Una volta terminato il ciclo si viene reindirizzati alla sezione '*pht*' dove verrà scattata la foto.

La sezione '*pht*' ha lo scopo di scattare la foto, per farlo è sufficiente cambiare il bit numero 3 della sequenza e metterlo a 1.
Per modificare il bit utilizzo una 'ori' tra la sequenza di bit su cui sto lavorando e un numero in esadecimale che abbia tutti zeri tranne un uno nella posizione numero 3 (0x0008 = 0000 0000 0000 1000) l'operazione sarà la seguente:
0000 0000 0000 0000 | 0000 0000 0000 1000 = 0000 0000 0000 1000
In questo modo avrò modificato il bit numero 3 della sequenza, questa modifica mi rappresenta lo scatto della macchina fotografica.

Lo scatto dura 500ms, quindi è necessaria l'introduzione di un secondo ciclo 'perdi tempo', a questo scopo al termine della modifica del bit numero 3 si trova un'istruzione di salto incondizionato che ci porterà ad un secondo ciclo perditempo con apposita costante anch'essa dichiarata a inizio programma.

La seconda sezione di perditempo (*perdiT1*) è analoga alla precedente, l'unica variante è la costante `$s5` al posto di `$s6`. Terminati i 500ms si ha lo spegnimento della macchina fotografica e il conseguente re indirizzamento alla sezione 'cod_1', quindi all'inizio del codice per l'attesa della macchina successiva.

DOCUMENTAZIONE TECNICA – TEST PC SPIM:

premessa:

Dopo aver scritto il codice in un file di testo lo salviamo con estensione *.s* oppure *.asm*, successivamente apriamo il file da PC-SPIM.
Per motivi di semplificazione durante la parte di test con PC SPIM abbiamo scelto di forzare durante l'esecuzione i valori originali dei registri $s1, $s2, $s3, $s5 e $s6 (che contenevano i valori di count per le velocità rispettive di 90km/h, 100km/h e 110km/h e i numeri di decremento dei cicli 'perdi tempo') a dei valori inferiori e più facili da gestire:

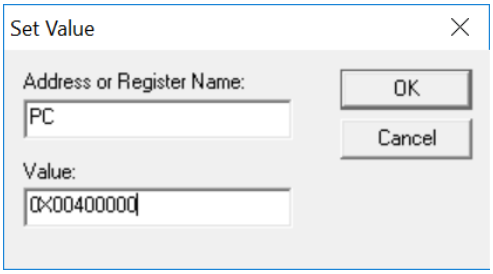
$s1 = n1 → e (14)
$s2 = n2 → a (10)
$s3 = n3 → 4

Quindi:
- *V < 90km/h* si identifica con **count > 14**
- *90km/h < V < 100km/h* si identifica con **10 < count < 14**
- *100km/h < V < 110km/h* si identifica con **4 < count < 10**
- *V > 110km/h* si identifica con **count < 4**

Mentre i registri $s5 e $s6 assumono i valori di *4* e *10*(a).

esecuzione dei test:

Dopo aver caricato il codice in PC SPIM come prima cosa setto il valore del Program Counter (PC) a 0x00400000 [Simulator -> Set Value]



Successivamente, premendo il tasto f10 eseguo istruzione per istruzione e testo il funzionamento del mio codice.
All'avvio tutti i valori dei registri vengono settati correttamente:

```
R0  (r0) = 00000000  R8  (t0) = 10010000  R16 (s0) = 00000000  R24 (t8) = 00004000
R1  (at) = 00000000  R9  (t1) = 00000000  R17 (s1) = 0000000e  R25 (t9) = 00000000
R2  (v0) = 00000000  R10 (t2) = 00000000  R18 (s2) = 0000000a  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000004  R27 (k1) = 00000000
R4  (a0) = 00000000  R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000
R5  (a1) = 00000000  R13 (t5) = 00000000  R21 (s5) = 0000000a  R29 (sp) = 7ffff60c
R6  (a2) = 7ffff614  R14 (t6) = 00000000  R22 (s6) = 00000004  R30 (s8) = 00000000
R7  (a3) = 00000000  R15 (t7) = 00008000  R23 (s7) = 00000000  R31 (ra) = 00000000
```

Questi settaggi derivano dal segmento di codice etichettato come '*cod_1*' all'interno del quale si hanno tutte le dichiarazioni delle varie variabili e costanti.
Dato che sono costanti a 32 bit sono state inserite tramite una `lui` e una `ori`, PC SPIM di conseguenza ha caricato prima i 16 bit più significativi (passati attraverso la `lui`) e successivamente i 16 bit meno significativi (passati dalla `ori`).

Nella documentazione verranno eseguiti i test affrontando casistica per casistica:

CASO 1: non passa nessuna auto, quindi i sensori non sono attivi.

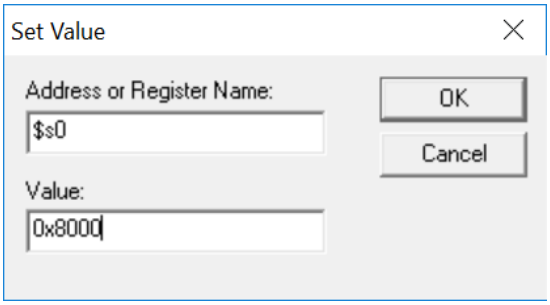
Andando avanti con l'esecuzione del codice potremo notare che il programma non esce dal ciclo '*loop1*' perché questo è un ciclo che controlla che il primo sensore sia settato a 1 e i sensori sono entrambi disattivati dato che nessuna auto è passata.

```
[0x0040004c]    0x120f0002  beq $16, $15, 8 [loop-0x0040004c]; 40: beq $s0, $t7, loop
[0x00400050]    0x08100013  j 0x0040004c [loop1]             ; 41: j loop1
```

Per come è scritto il codice il ciclo continuerà ad essere eseguito fintanto che non rileverà il passaggio di un'auto.

CASO 2: passa un'auto che va ad una velocità nella norma (minore o uguale a 90km/h).

Setto il valore del bit 15 (primo sensore) a 1 (il che indica che è passate un'auto)

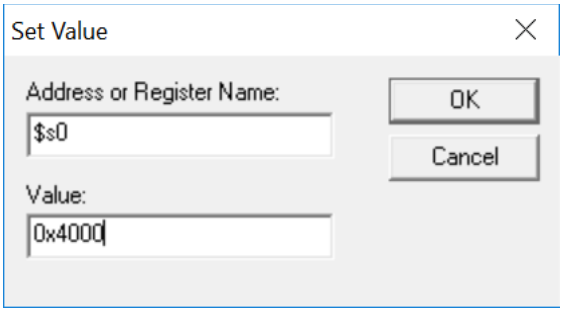


In questo modo il registro $s0 ha il bit più significativo settato a 1 `R16 (s0) = 00008000`

Procedendo con l'esecuzione del codice PC SPIM uscirà dal ciclo 'loop1' per entrare nel ciclo 'loop'

```
[0x00400054]    0x22940001  addi $20, $20, 1               ; 43: addi $s4, $s4, 1
[0x00400058]    0x12180002  beq $16, $24, 8 [check-0x00400058]; 44: beq $s0, $t8, check
[0x0040005c]    0x08100015  j 0x00400054 [loop]            ; 45: j loop
```

Qui andrà avanti ad incrementare 'count' fintanto che non setterò il bit numero 15 a 1. Dato che sto testando la casistica V < 90km/h incrementerò 'count' un numero di volte superiore a 14 `R20 (s4) = 00000012`
Setto il bit 15 a 1 ed esco dal ciclo



All'uscita dal ciclo viene effettuato il controllo sulla velocità (sezione check), in questo caso dato che la mia velocità è minore di 90km/h non effettuo altri controlli, salto alla sezione 'fine' che mi rimanda all'inizio del codice azzerando tutti i registri modificati

```
[0x00400060]    0x0234582a  slt $11, $17, $20              ; 47: slt $t3, $s1, $s4
[0x00400064]    0x34010001  ori $1, $0, 1                  ; 48: beq $t3, 1, fine
[0x00400068]    0x102b0002  beq $1, $11, 8 [fine-0x00400068]
[0x0040006c]    0x11600002  beq $11, $0, 8 [vel-0x0040006c] ; 49: beq $t3, $zero, vel
[0x00400070]    0x08100002  j 0x00400008 [cod_1]           ; 51: j cod_1
```

CASO 3: passa un'auto che va ad una velocità maggiore di 90km/h e minore di 100km/h.

Come per il caso precedente setto il valore di PC, il valore di $s0 e incremento 'count' in un numero compreso tra 10 e 14 per far risultare la velocità compresa tra 90km/h e 100km/h.

Incremento 'count' a 11(b) `R20 (s4) = 0000000b`

Andando avanti con l'esecuzione arriviamo al ciclo in cui si chiede se la velocità è maggiore o minore di 90km/h, nel caso precedente a questo ciclo siamo stati reindirizzati alla sezione di codice etichettata 'fine' dato che la nostra velocità era minore di 90km/h, ora è maggiore quindi verremo rimandati ad un ulteriore controllo per sapere in quale casistica dell'infrazione si trova la nostra velocità.

```
[0x00400060]    0x0234582a  slt $11, $17, $20                ; 47: slt $t3, $s1, $s4
[0x00400064]    0x34010001  ori $1, $0, 1                    ; 48: beq $t3, 1, fine
[0x00400068]    0x102b0002  beq $1, $11, 8 [fine-0x00400068]
[0x0040006c]    0x11600002  beq $11, $0, 8 [vel-0x0040006c]  ; 49: beq $t3, $zero, vel
```

Arrivo a quello che sarà l'ultimo controllo per la velocità in questione: la sezione 'vel' in cui controllo se la velocità è maggiore o minore di 100km/h, in questo caso è minore quindi verrò reindirizzata alla sezione 'fine_1'

```
[0x00400074]    0x0254682a  slt $13, $18, $20                ; 56: slt $t5, $s2, $s4
[0x00400078]    0x34010001  ori $1, $0, 1                    ; 57: beq $t5, 1, fine_1
```

Questa è la sezione finale per la casistica in cui la velocità sia compresa tra 90km/h e 100km/h, quindi i bit numero 6 e 7 vanno settati a 0 e 1:

```
[0x00400094]    0x36100080  ori $16, $16, 128               ; 64: ori $s0, $s0, 0x0080
[0x00400098]    0x0810002b  j 0x004000ac [perdiT]           ; 65: j perdiT
```

La sequenza di bit risulta essere così al termine delle precedenti istruzioni:

`R16 (s0) = 00004080`

Arrivata a questo punto dovrò scattare la foto, dopo aver sostituito i bit della velocità verrò quindi reindirizzata al ciclo perditempo.

Questo ciclo decrementerà $s6 fino a farlo arrivare a 0 (allo scopo di far passare 1 secondo prima di scattare la foto), terminato questo decremento verremo reindirizzati alla sezione 'pht' in cui verrà effettivamente scattata la foto modificando il bit numero 3 del registro $s0 (che contiene la sequenza di bit)

```
[0x004000ac]    0x22d6ffff  addi $22, $22, -1               ; 76: addi $s6, $s6, -1
[0x004000b0]    0x12c00002  beq $22, $0, 8 [pht-0x004000b0] ; 77: beq $s6, $zero, pht
[0x004000b4]    0x0810002b  j 0x004000ac [perdiT]           ; 78: j perdiT
[0x004000b8]    0x36100008  ori $16, $16, 8                 ; 80: ori $s0, $s0, 0x0008
[0x004000bc]    0x08100030  j 0x004000c0 [perdiT1]          ; 81: j perdiT1
```

Il bit numero 3 del registro $s0 viene settato a 1: `R16 (s0) = 00004088`

Dopo di che passeranno i 500ms necessari allo scatto della foto, vado quindi alla sezione 'perdiT1' e decremento il contenuto del registro $s5 finché non diventa pari a 0, dopo di che ritorno a cod_1 (inizio del codice) e azzero tutti i valori aspettando la macchina successiva

```
[0x004000c0]    0x22b5ffff  addi $21, $21, -1               ; 83: addi $s5, $s5, -1
[0x004000c4]    0x12a0ffff  beq $21, $0, -4 [perdiT1-0x004000c4]; 84: beq $s5, $zero, perdiT1
[0x004000c8]    0x08100002  j 0x00400008 [cod_1]            ; 85: j cod_1
```

La stringa di bit finale per questa tipologia di infrazione è:

che equivale a 0100 0000 1000 1000

* Macchina Fotografica
* Bit Velocità

CASO 4: passa un'auto che va ad una velocità maggiore di 100km/h e minore di 110km/h.

Per avere una velocità compresa tra 100km/h e 110km/h inserisco un valore di 'count' che sia compreso tra 4 e 10

```
R20 (s4) = 00000006
```

Fino alla sezione di codice denominata 'vel' la situazione è analoga a quella del caso precedente, quando arriviamo a questa porzione di codice però non verremo reindirizzati ad una sezione di termine ma ad una sezione ('vel_1') che effettua un ulteriore controllo per sapere se la mia velocità è maggiore o minore di 110 km/h

```
[0x00400084]    0x0274682a  slt $13, $19, $20                ; 60: slt $t5, $s3, $s4
[0x00400088]    0x34010001  ori $1, $0, 1                    ; 61: beq $t5, 1, fine_2
```

In questo caso dato che la nostra velocità è minore di 110 km/h andremo nella sezione 'fine_2', qui come nel caso precedente verranno modificati i bit 6 e 7 settandoli a 1 e 0

```
[0x0040009c]    0x36100040  ori $16, $16, 64                 ; 68: ori $s0, $s0, 0x0040
[0x004000a0]    0x0810002b  j 0x004000ac [perdiT]            ; 69: j perdiT
```

Terminata questa sezione di codice salto al ciclo perditempo che in modo assolutamente analogo al precedente mi scatterà la foto.

La stringa di bit finale per questa tipologia di infrazione è: `R16 (s0) = 00004048`

che equivale a 0100 0000 0100 1000
* Macchina Fotografica
* Bit Velocità

CASO 5: passa un'auto che va ad una velocità maggiore di 110km/h

Per avere una velocità compresa tra 100km/h e 110km/h inserisco un valore di 'count' che sia minore di 4

```
R20 (s4) = 00000002
```

Il codice viene eseguito nello stesso modo rispetto ai casi precedenti fino alla sezione 'vel_1' in cui si viene reindirizzati in due sezioni differenti in base alla velocità, prima avevamo una velocità minore di 110 km/h, quella che stiamo analizzando ora è invece superiore a 110km/h, di conseguenza verrà eseguita una porzione di codice ancora differente rispetto a quella del caso 4

```
[0x00400084]    0x0274682a  slt $13, $19, $20                ; 60: slt $t5, $s3, $s4
[0x00400088]    0x34010001  ori $1, $0, 1                    ; 61: beq $t5, 1, fine_2
[0x0040008c]    0x102d0004  beq $1, $13, 16 [fine_2-0x0040008c]
[0x00400090]    0x11a00005  beq $13, $0, 20 [fine_3-0x00400090]; 62: beq $t5, $zero, fine_3
```

Arriviamo alla sezione 'fine_3' che è la sezione termine dell'ultima casistica, qui setteremo i bit 6 e 7 a 1 e 1

```
[0x004000a4]    0x361000c0  ori $16, $16, 192                ; 72: ori $s0, $s0, 0x00C0
[0x004000a8]    0x0810002b  j 0x004000ac [perdiT]            ; 73: j perdiT
```

Terminata la modifica dei bit verrà avviato il ciclo perditempo e scattata la foto.

La stringa di bit finale per questa tipologia di infrazione è: `R16 (s0) = 000040c8`

che equivale a 0100 0000 1100 1000
* Macchina Fotografica
* Bit Velocità