#### **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:

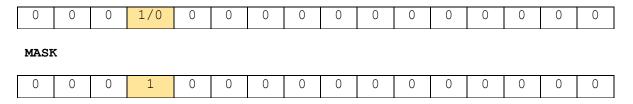
l'idea è quella di caricare nella MASK il valore che serve a noi per poi fare un and bit a bit tra la maschera e i bit di IN\_OUT, visto che l'operazione logica and da in uscita 1 solo nel caso in cui entrambi i bit valgono 1, cosi facciamo il programma ritornare al ciclo 1 per ricontrollare il sensore 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



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
MASI	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 500 Mhz è in grado di eseguire 500 milioni 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 **0,04s** 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 \$55 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:

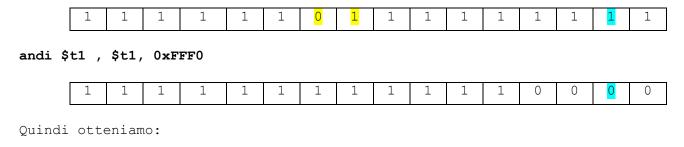
- 1. Conoscenza che la macchina è passata, ed a quale velocità (indicazioni contenute nella half-word nel registro \$t0
- 2. Attesa 500ms
- 3. Scatto della foto con impulso pari a 100ms
- 4. Terminazione del processo: riporto tutto alla condizione iniziale

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 <a href="bit1">bit1</a>). 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

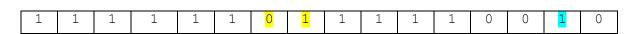




Così facendo otteniamo i primi 4 bit che sono pari a 0. Fotocamera pronta per scattare. Eseguo  $\tt ori~\$t1,\$t1,0x0002$ 



Quindi otteniamo:



Così facendo abbiamo settato il bitl 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	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

# 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	$\cap$	1	1	1	1	1	$\cap$	$\cap$		$\cap$
							U						U	U	U	U
L																

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