Condiciones

HS — LS	EQ — HE	Ver condicionales y los saltos que usa.
HI — LO	MI — PL	
GT — LT	VS — VC	
GE — LE	CS — CC	

Subs r1, r2

Si: r1 < r2 → LO

Si: r1 ≥ r2 → HS

Si: r1 == r2 → EQ

cmp. r1, r2 /compara

movb r1, r2 /copia r2 en r1 si la comparación dió que r1 < r2

cmp r1, r2
b ___ es igual al goto y va a la etiqueta

```
cmp r1, r2
bhs saltar
mov r1, r2
saltar: .....
        .....
        .....
```

Sacar el valor absoluto de un nro.

```
        cmp  r1, #0
        bpl  pos
        mov  r2, #0
        sub  r2, r1
        mov  r0, r2
pos:
```

17/05/16

```
cmp r0, #0
bpl pos
```

```
mov r1, #0
sub r1, r0
mov r0, r1
pos:
    :
```

```
mov r1, #0   → rsb r0, #0
sub r0, r1, r0
sub r0, #0, r0
sub A, B  → A-B
rsb A, B  → B-A
```

```
if (r0 > 20)
{
  r0 = 20;
}

cmp r0, #20
bls salto
mov r0, #20
salto
```

```
if (r0 > 10 && r0 < 20)

Si:
No:

cmp r0, #10
ble No
cmp r0, #20
bge No
Si
    b cont
No ——
   ——
```

```
if (r0 == 10 || r0 == 20)

        cmp r0, #10
        beq igual
        cmp r0, #20
        beq igual
        b cont
  igual: ——
         ——
  cont
```

```
if (r0 < 0)
{
    r1 = 1; }
else
{
 r1 = 0;
}

cmp r0, #0
bge no menor
mov r1, #1
b continuar
no menor: mov r1, #0
continuar
```

```
if (r1 != 0)

igual:
distinto:

cmp r1, #0
bne distinto
mov
b cont
distinto
cont
```

húsares

13/09/16

vec1: .asciz "Hola, Hola Mundo"
vec2: asciz "Hola"

Comparion de dos cadenas.

```
        ldr r1, =vec1
        ldr r2, =vec2

        ldr r3, [r2], #1
otro:   ldr r4, [r1], #1

        cmp r3,r4
        beq iguales

        b otro
```

```
otro:   ldrb r3, [r1], #1
        ldrb r4, [r2], #1
        cmp r3,r4
        bne distintos
        cmp r3, #0
        beq iguales
        b   otro
```

↓

haciendo la comparación con subrrotiña de cadena

```
otro:   ldrb r3, [r0], #1
        ldrb r4, [r2], #1
        cmp  r4, #0
        beq  iguales
        bne  distintos
        b    otro.
```

Programa principal

```
        ldr r0, =vec1
continuar: ldr r2, =vec2
        mov r0, r1

        b otro
iguales:  add r5, #1
distintos: add r1, #1
        ldrb r3, [r1]
        cmp r3, #0
        beq fin
        b continuar
```

con funcion

→ b otro →

```
bl comparar
cmp r0, #0
addeq r5, #1
```

↓

Ahora la subrrutina con una función

```
comparar : mov r0, #0
   otro:   ldrb r3, [r0], #1
           ldrb r4, [r2], #1
           cmp  r3, #0
           beq  fin_compara
           cmp  r3,r4
           beq   otro
           mov   r0, #1
fin_compara: mov pc, lr
```

En las funciones se usan los primeros 4 registros para devolver valores

---

```
comparar : push {r1,r2,r3,r4}

           mov r2, r0
           mov r0, #0
otro :     ldrb r3, [r2], #1 @ vec1
           ldrb r4, [r1], #1 @ vec2
           cmp  r4, #0
           beq  fin_compara
           cmp  r3,r4
           beq   otro
           mov   r0, #1
fin_compara: pop {r1,r2,r3,r4}
           mov pc, lr
```

```
           ldr r2, =vec1
           ldr r1, =vec2

continuar: mov r0, r2
           bl  comparar
           cmp r0, #0
           addeq r5, #1
           add  r2, #1
           ldrb r3, [r2]
           cmp r3, #0
           bne continuar
           :
           :
```

if (r0 ≤ 50 && r2 < 30)

```
comp r0, #50
blt  No
cmp  r2, #30
bge  No
Si: ...
    ...
    b cout
No: ...
    ...
```

if [(r0 ≥ 'A' && r0 ≤ 'Z') || (r0 ≥ 'a' && r0 ≤ 'z')]

```
comp r0, #'A'
blo  cond,2
comp r0, #'Z'
bhi  cond2
b    Si
cond2 comp r0, #'a'
blo  No
comp r0, #'z'
bhi  No

Si: ...
    ...
    b cout
No: ...
    ...
cout:
```

suma add r1, r2 { para hacer
mov pc, lr { funciones
y volver
con el program
counter

30/08/16

Comparar dos cadenas de texto devolver en r0 = 1 si son iguales
r0 = 0 si son distintas

cadena: .byte 'h','o','l','a','0', 0.

cadena: .asci "hola0"

cadena: .asciz "hola0"

cadena2: asciz "Hola0".

```
bucle :  cmp  r0, 0
         beq  salir
         LDRIA R0
         LDRIA R1
         Cmp  R0,R1

         b bucle
salir
```

13/09/16

falta clase anterior

27/09/16

Dado 2 vect por cada letra ver cuantas veces se repiten en cada vector

```
         .ltorg      → donde va a guardar el número si es muy extenso
vect1: .asciz "Hola mundo"

vect 2: .byte 'H','o','m',0.
         .balign 2   → alinea la memoria para half word o
cout : .hword 0,0,0     para 4 bytes así guardar a principio de memoria
         .balign4 →
```

así se graban las cadenas

DS1				DS
a	l	o	H	
m	u	m	-	
H	nul	o	d	
	nul	m	o	

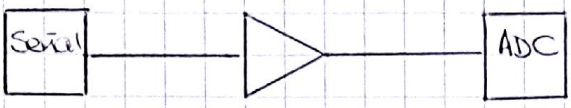
↓

Manera 1 Tomo el primer elemento de vec2 y comparar lo con todo vec1

```
         ldr r1,=vec1
         ldr r2,=vec2
         ldr r3,=cout

otro:    ldr r4,[r2],#1
         cmp r4,#0
         beq salir
         mov r0,#0
         ldr r1,vec1
otro_bucle: ldrb r5,[r1],#1
         cmp r5,#0
         beq salir_bucle
         cmp r4,r5
         addeq r0,#1
         b otro_bucle

salir_bucle: strh r0,[r3],#2

         b otro:
```

11/10/16

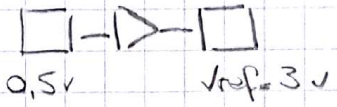
Adc.



$\pm 10\ mV \rightarrow 0 - 20\ mV \rightarrow V_{ref} = 5V$
$G = 250$

$G = \frac{V_{ref}}{V_s}$

Si tenemos una señal de 0,5 V



$V_{pp} = 1\ mV$

$pasos = \frac{V_{pp}}{V_{resolución}} = \frac{1V}{1mV} = 1000$

$2^n = 1000 \quad n = \frac{ln(1000)}{ln(2)} = 9,96 \approx 10$ (resolución)

Cálculo de la ganancia.

$G = \frac{V_{ref}}{V_{pp}} = \frac{3}{1} = 3$

Para una correcta resolución y salto la ganancia se calcula diferente

$G = \frac{V_{lsb}}{V_{res}} = \frac{\frac{V_{ref}}{1024}}{1mV} = \frac{\frac{3V}{1024}}{1mV} = \frac{2,92\ mV}{1\ mV} = 2,92$

Error total de un sistema como estos

$E_T = E_{ADC} + E_G + E_{riple}$

la suma de estos no debe ser mayor al error E_{ADC}

$E_r = \frac{V_{ref}}{2^n \cdot 4}$

$$EG = \frac{\frac{Vref}{2^n \cdot 4}}{Vref} = \frac{1}{2^n \cdot 4}$$

Adecuación del tiempo o ftec.

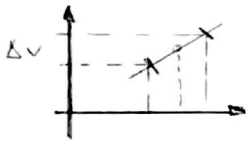
$Fs = 1\ KHz$

$F = 1\ KHz \times 5\ veces = 5\ KHz$

El tiempo de conversión

$$tc = \frac{1}{5\ KHz} = 200\ \mu s$$

Ahora el muestreo de sampleo



$v = V \operatorname{Sen} \omega t$

$$\frac{dv}{dt} = V\ \omega . \operatorname{Cos} \omega t$$

$$\frac{\Delta v}{ts} = V . 2\pi F$$

$$\Delta v = V . ts\ 2\pi F$$

$$e = \frac{\Delta v}{2.V} = ts . \pi F < \frac{1}{2^n . 2}$$

$$ts < \frac{1}{2^n . 2 . \pi . F}$$

$p/n = 12$ y $F = 1\ KHz$

$ts < 39\ ns$

Buscar el pto. x,y más alejado del origen

Tenemos un vector

Vec: .hword -10,20,-5,10,8,30

Si no tengo la raíz cuadrada no importa porque

$20 < 30$
$\sqrt{20} < \sqrt{30}$ } me sigue comparando

```
        ldr  r0, =vec
sig     ldr  r1, [r0],
        mul  r2 r1, r1
        ldr  r0, #1
        mul  r3, r1, r1
        add  r4, r2, r3
        ldr  r4, #1
```

str r4, [ ]

25/10/16

Realizar un programa que da un vector de 200 elem. word. y un vector de 50 elem del mismo tipo, cuente y sume los valo. del primer vector siempre que estos se encuentren en el seg.

vect 1: 10,6,7,8,9,6,9,3,5,2

vect 2: 6,8,9,1,5,4

```
        ldr  r1 = vec1
        ldr  r2 = vec 2
        mov  r3, #0      } acumular
        mov  r4 =, #0    } pos.
        mov  r5, #0
bucle 1 ldr  r6 [r1], #4
        add  r4, #1

        ldrb r7, [r2], #1
        add  r5, #1

        cmp  r6, r7
        add eq r3, r6
        mul  r8, r5, #4
        ldr  r6, #-r8
        mov  r5, #0
```

} bucle 1

```
beq     buck1
cmp     r5, #50
beq     buck1
cmp     r4 #200
beq     salir
```

(2) Un vector .word de punteros está cargado con punteros de cadenas de caracteres, contar cuantas de ellas tienen la letra en mayúsculas.

```
        mov r4, #0
        mov r5, #0
        ldr r2, = vec1
        ldr r1, = vec2
        mov r3, #200
otro:   ldr r0, [r2], #4
        mov r6, r0
        bl buscar
        cmp r0, #0
        bne cont
        add r4, r6
        add r5, #1
cont.   subs r3, #1
        bne otro
```

```
buscar:         push {r1-r3}
                mov r3, #50
buscar_otro:    ldr r2, [r1], #4
                cmp r2, r0
                beq buscar_si
                subs r3, #1
                bne buscar_otro
                mov r0, #1
                b buscar_salir
buscar_si:      mov r0, #0
buscar_salir:   pop {r1-r3}
                mov pc, lr
```