



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNM
TECNOLOGICO NACIONAL DE
MÉXICO



Tecnológico Nacional de México

Instituto Tecnológico de Tijuana

Subdirección Académica

Depto. De Sistemas y Computación

Semestre Febrero – Junio 2022

Ingeniería en Sistemas Computacionales

Materia: Lenguajes de Interfaz SCC-1014SC6A

Maestro: Rene Solís Reyes

Unidad: 2

Tarea: 2.5 Avance Cuadernillo

Alumno: Dueñas Nuñez Alan Gabriel

Número De Control: 19211630

Fecha de entrega: 5/abril/2022

Ejercicio 1.3

Si interpretamos el resultado como byte	
binario	hexa
11110010	F2
Si interpretamos el resultado como palabra (32 bits)	
binario	hexa
11110010	F2
Ídem, pero si no hubiésemos extendido los signos (ldrb en lugar de ldrsb)	
binario	hexa
11110010	F2

Al usar **ldrsb** los registros tras 5 líneas ejecutadas son:

```
(gdb) start
Temporary breakpoint 1 at 0x103d0
Starting program: /home/GabrielXIX/intro2

Temporary breakpoint 1, 0x000103d0 in main ()
(gdb) stepi
0x000103d4 in main ()
(gdb) stepi
0x000103d8 in main ()
(gdb) stepi
0x000103dc in main ()
(gdb) stepi
0x000103e0 in main ()
(gdb) stepi
0x000103e4 in main ()
(gdb) disassemble
Dump of assembler code for function main:
   0x000103d0 <+0>:    ldr     r1, [pc, #16]    ; 0x103e8 <main+24>
   0x000103d4 <+4>:    ldrsb   r1, [r1]
   0x000103d8 <+8>:    ldr     r2, [pc, #12]    ; 0x103ec <main+28>
   0x000103dc <+12>:   ldrsb   r2, [r2]
   0x000103e0 <+16>:   adds    r0, r1, r2
=> 0x000103e4 <+20>:   bx      lr
   0x000103e8 <+24>:   andeq   r1, r2, r4, lsr #32
   0x000103ec <+28>:   andeq   r1, r2, r8, lsr #32
End of assembler dump.
(gdb) i r r1
r1                0x32                50
(gdb) i r r2
r2                0xffffffc0         4294967232
(gdb) i r r0
r0                0xfffffffff2       4294967282
(gdb) []
```

Pero al cambiar a **ldrb** (sin extender signos) son:

```
(gdb) start
Temporary breakpoint 1 at 0x103d0
Starting program: /home/GabrielXIX/intro2

sTemporary breakpoint 1, 0x000103d0 in main ()
(gdb) stepi
0x000103d4 in main ()
(gdb) stepi
0x000103d8 in main ()
(gdb) stepi
0x000103dc in main ()
(gdb) stepi
0x000103e0 in main ()
(gdb) stepi
0x000103e4 in main ()
(gdb) disassemble
Dump of assembler code for function main:
    0x000103d0 <+0>:    ldr     r1, [pc, #16]    ; 0x103e8 <main+24>
    0x000103d4 <+4>:    ldrb    r1, [r1]
    0x000103d8 <+8>:    ldr     r2, [pc, #12]    ; 0x103ec <main+28>
    0x000103dc <+12>:   ldrb    r2, [r2]
    0x000103e0 <+16>:   adds    r0, r1, r2
=> 0x000103e4 <+20>:   bx      lr
    0x000103e8 <+24>:   andeq   r1, r2, r4, lsr #32
    0x000103ec <+28>:   andeq   r1, r2, r8, lsr #32
End of assembler dump.
(gdb) i r r1
r1                0x32                50
(gdb) i r r2
r2                0xc0                192
(gdb) i r r0
r0                0xf2                242
(gdb) []
```

Ejercicio 1.4

Repite el ejercicio anterior, pero ahora comprobando el resultado de los flags con lo que habías calculado en el Ejercicio 1.2. ¿Qué ocurre?

Al cambiar la instrucción de **add** por **adds** el registro **cpsr** es el siguiente:

```
(gdb) i r cpsr
cpsr      0x80000010      -2147483632
(gdb) □
```

Entonces 8 -> 0b1000, por lo tanto, los flags están como: **N=1**, Z=0, C=0, V=0.

Ejercicio 1.5

Supón que tienes dos variables de tamaño 1 byte, **var1** y **var2**, con los valores 11110000_b y 10101010_b . Calcula el resultado de hacer una operación AND y una operación OR entre las dos variables.

Variable	Valor (binario)		Valor (hexadecimal)		Valor (binario)
var1	11110000		F0		11110000
var2	10101010		AA		10101010
<hr/>					
var1 AND var2	10100000	A0	var1 OR var2	11111010	FA
	binario	hexa		binario	hexa

Resultados de los registros **r0=AND** y **r1=OR**

```
Dump of assembler code for function main:
0x000103d0 <+0>:  mov     r2, #240          ; 0xf0
0x000103d4 <+4>:  mov     r3, #170          ; 0xaa
0x000103d8 <+8>:  and     r0, r2, r3
0x000103dc <+12>: orr     r1, r2, r3
=> 0x000103e0 <+16>: mvn     r4, r0
0x000103e4 <+20>: mov     r0, #-2147483648   ; 0x80000000
0x000103e8 <+24>: tst     r0, #-2147483648   ; 0x80000000
0x000103ec <+28>: tst     r0, #1073741824    ; 0x40000000
0x000103f0 <+32>: bx      lr
End of assembler dump.
(gdb) i r r0
r0      0xa0      160
(gdb) i r r1
r1      0xfa      250
```

Ejercicio 1.6

El resultado de la instrucción `and` está en `r0`. ¿Cuál será el resultado de hacer un complemento a uno del mismo?

	binario	hexa
<code>r0</code>	10100000	A0
	binario	hexa
<code>~r0</code>	01011111	5F

Ejecuta con el `gdb` la instrucción `mvn r4, r0` y comprueba tu respuesta.

Resultado de instrucción `mvn r4, r0`

```
Dump of assembler code for function main:
0x000103d0 <+0>:  mov     r2, #240      ; 0xf0
0x000103d4 <+4>:  mov     r3, #170      ; 0xaa
0x000103d8 <+8>:  and     r0, r2, r3
0x000103dc <+12>: orr     r1, r2, r3
0x000103e0 <+16>: mvn     r4, r0
=> 0x000103e4 <+20>: mov     r0, #-2147483648 ; 0x80000000
0x000103e8 <+24>: tst     r0, #-2147483648 ; 0x80000000
0x000103ec <+28>: tst     r0, #1073741824 ; 0x40000000
0x000103f0 <+32>: bx      lr
End of assembler dump.
(gdb) i r r0
r0                0xa0                160
(gdb) i r r4
r4                0xffffffff5f        4294967135
```

Ejercicio 1.7

La instrucción `tst` hace la operación `and` entre un registro y una máscara y sólo actúa sobre los flags. Complimenta las casillas en blanco, teniendo en cuenta que el flag Z se pone a uno cuando el resultado de la `and` es cero, y se pone a cero en caso contrario. Para simplificar indicamos sólo los 16 bits menos significativos del registro `r0`.

	binario	hexa
r0	10000000000000000000000000000000	80000000
tst r0, #0x80000000		Z? 0
tst r0, #0x40000000		Z? 1

Comprueba tus respuestas con ayuda del `gdb`, y examina el resto de flags, observa qué ocurre con el flag N (flag de signo).

Primera instrucción `tst`: los flags quedaron como **N=1**, Z=0, **C=1**, V=0

```
0x000103e8 <+24>:  tst    r0, #-2147483648 ; 0x80000000
=> 0x000103ec <+28>:  tst    r0, #1073741824 ; 0x40000000
0x000103f0 <+32>:  bx      lr
End of assembler dump.
(gdb) i r cpsr
cpsr      0xa0000010      -1610612720
```

Segunda instrucción `tst`: los flags quedaron como N=0, **Z=1**, C=0, V=0

```
0x000103e8 <+24>:  tst    r0, #-2147483648 ; 0x80000000
0x000103ec <+28>:  tst    r0, #1073741824 ; 0x40000000
=> 0x000103f0 <+32>:  bx      lr
End of assembler dump.
(gdb) i r cpsr
cpsr      0x40000010      1073741840
```

Subiendo los archivos a github con clone, add, commit y push

git clone

```
GabrielXIX@raspberrypi:~ $ git clone https://github.com/tectijuana/avancep21-27-GabrielXIX
Cloning into 'avancep21-27-GabrielXIX'...
Username for 'https://github.com': GabrielXIX
Password for 'https://GabrielXIX@github.com':
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 20 (delta 4), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (20/20), done.
```

Copiando archivos a la carpeta y usando git add

```
GabrielXIX@raspberrypi:~/avancep21-27-GabrielXIX $ cp /home/GabrielXIX/intro1.s /home/GabrielXIX/avancep21-27-GabrielXIX
GabrielXIX@raspberrypi:~/avancep21-27-GabrielXIX $ cp /home/GabrielXIX/intro2.s /home/GabrielXIX/avancep21-27-GabrielXIX
GabrielXIX@raspberrypi:~/avancep21-27-GabrielXIX $ cp /home/GabrielXIX/intro3.s /home/GabrielXIX/avancep21-27-GabrielXIX
GabrielXIX@raspberrypi:~/avancep21-27-GabrielXIX $ ls
intro1.s intro2.s intro3.s README.md
GabrielXIX@raspberrypi:~/avancep21-27-GabrielXIX $ git add intro1.s
GabrielXIX@raspberrypi:~/avancep21-27-GabrielXIX $ git add intro2.s
GabrielXIX@raspberrypi:~/avancep21-27-GabrielXIX $ git add intro3.s
```

git commit -m

```
GabrielXIX@raspberrypi:~/avancep21-27-GabrielXIX $ git config --global user.name "GabrielXIX"
GabrielXIX@raspberrypi:~/avancep21-27-GabrielXIX $ git commit -m "Cambios en el repo de avance 2.5"
[main b8fee7f] Cambios en el repo de avance 2.5
Committer: GabrielXIX <GabrielXIX@raspberrypi>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

3 files changed, 61 insertions(+)
create mode 100644 intro1.s
create mode 100644 intro2.s
create mode 100644 intro3.s
```

git push origin main

```
GabrielXIX@raspberrypi:~/avancep21-27-GabrielXIX $ git push origin main
Username for 'https://github.com': GabrielXIX
Password for 'https://GabrielXIX@github.com':
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 1.08 KiB | 551.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/tectijuana/avancep21-27-GabrielXIX
3f06151..b8fee7f main -> main
```