102685 Arquitectura de Computadors i Perifèrics PRÀCTICA Nº 3 CURS 2017/2018

INCLUSIÓ DE CODI ASSEMBLADOR DINS DE C I OPERADORS BINARIS

Objectius de la pràctica

En aquesta sessió de pràctiques tractarem els següents temes:

- 1) Com incloure instruccions assemblador del processador Cortex M0 en el codi del llenguatge de programació C.
- 2) Com funcionen els operadors binaris que ofereix el llenguatge C.

Inclusió d'instruccions assemblador en el codi C

En les dues pràctiques anteriors hem utilitzat el llenguatge assemblador del processador Cortex M0 per generar els seus codis executables (utilitzant el Code Warrior). En entorns reals que utilitzen plaques com la que treballem en pràctiques (coneguts com a sistemes encastats), els llenguatges de programació utilitzats en els seus codis fonts solen ser els d'alt nivell com el C. El entorn de treball del CodeWarrior no n'és una excepció i permet utilitzar el llenguatge C com a codi font dels executables del processador Cortex M0. Però hi ha ocasions (per optimitzar velocitat o per accedir a cert recurs) que és necessari incloure instruccions assemblador dins del codi C. Per fer això, aquest llenguatge d'alt nivell ofereix la sentència especial *asm* ("instrucció assemblador ARM"). Per exemple, el següent codi C inclou la instrucció assemblador d'ARM *mov R1,#3*:

```
int main (void) {
    asm ("mov R1,#3");
    return (1);
}
```

També es poden incloure múltiples sentencies assemblador amb una sola sentencia *asm* de les següents formes:

- 1) asm (" 1^a instrucció assemblador ARM ; 2^a instrucció assemblador ARM ; ... ; N instrucció assemblador ARM").
- 2) asm ("1ª instrucció assemblador ARM \n" "2ª instrucció assemblador ARM \n" ... "N instrucció assemblador ARM").

Per exemple, els següents codis C inclouen dues instruccions assemblador:

Operadors pel tractament de bits en C

Quan s'utilitza el llenguatge C en sistemes encastats, és molt usual haver de modificar o llegir certs bits d'alguna variable o d'algun registre, ja sigui del processador o d'entrada/sortida (els quals es veuran en pròximes sessions de pràctiques).

Per realitzar aquestes operacions, el llenguatge C disposa d'un conjunt d'operadors binaris, els quals són:

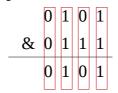
• ~: Complement a u, inverteix cada bit de l'operand. Per exemple:

$$2 << 4 = 32 => 00000010 << 4 = 00100000$$

• >> : Desplaçament a bits cap a la dreta. La sintaxi d'aquest operador és: expressió a desplaçar >> nombre a bits a desplaçar

Per exemple, si es vol desplaçar el número 32, 4 bits a la dreta es pot utilitzar:

• **&** : AND bit a bit. Per exemple: **5 & 7** = **5**



L'operació es realitza per cada bit dels dos operants, idependenment del resultat dels altres bits.

• \land : XOR (or exclusiva) bit a bit. Per exemple: $5 \land 7 = 2$

• |: OR bit a bit. Per exemple: 5 | 7 = 7

En el cas dels desplaçaments a la dreta i a l'esquerra, hem de tenir en compte les següents condicions:

- *Desplaçament a l'esquerra*: El bit més significatiu (el de més a l'esquerra) es perd i es posa un 0 en bit menys significatiu (el de més a la dreta).
- *Desplaçament a la dreta*: El bit menys significatiu es perd i es manté el signe pel més significatiu, això és, el nou bit més significatiu serà 0 si es un número positiu i 1 si és negatiu.

Exercici

Implementeu un programa en C que realitzi les següents accions:

- **1)** Defineixi un sencer *moviment*, inicialitzat amb qualsevol valor, i que utilitzant els operadors binaris, faci que tots els seus bits prenguin el valor 1.
- 2) Repeteixi (els cops que es vulgui) els dos moviments següents:
 - **2a)** Mogui, cada cert interval de temps, 4 bits a 0 del sencer *moviment*, de la dreta cap a l'esquerra de 4 en 4 bits:

	<u>Sencer moviment:</u>	
bits:	31	0
	1111111111111 0000	
	111111	11 0000 1111
	••••••	
	00001111	11111111

2b) Moviment en direcció contraria, Mogui els 4 bits a 0 de l'esquerra cap a la dreta de 4 en 4 bits:

Sencer moviment:		
bits:	310	
	0000 111111111111	
	1111 0000 11111111	
	•••••	
	11111111 1111 0000	

La funció que espera l'interval de temps, s'ha d'implementar amb C utilitzant bucles incloent la instrucció assemblador **nop** (amb la directiva **asm**).

Documentació

- Al Campus Virtual: Documents del codi assemblador del Cortex M0
- Inserció codi assemblador ARM en C: http://infocenter.arm.com/help/index.jsp? topic=/com.arm.doc.dui0472c/Cihdeeja.html
- Operadors binaris amb C: http://www.zator.com/Cpp/E4_9_3.htm