IIC2343 – Arquitectura de Computadores

AYUDANTÍA 4

Código Assembly

Constanza Osorio

Assembly

- Será lo que usaremos para poder programar en nuestro computador básico, de una forma más natural y comprensible para nosotros.
- Está dividido en 2 secciones: DATA y CODE.
- ¿Qué está haciendo el código de la derecha? (˚ − ˚)

```
1    DATA:
2    dummy   2
3
4    CODE:
5    MOV A,(dummy)
6    MOV B, dummy
7    ADD A,2
8    SUB A,3
```

Antes de seguir... ¿ Dónde pruebo mi código *assembly*?

- Emulador del curso, disponible en el repositorio de Github – sección *Releases*
- Para la instalación, revisen la *Wiki* del repositorio!
- La extensión del archivo es .txt
- Si tienen algún problema de instalación, usen las issues
- La versión actual del emulador es la **1.0.1**
- Recuerden que con los reportes de bugs pueden ganar décimas! °₊✧ ヽ(º ▿ º)ノ ✧₊°

¿Cuáles son las instrucciones soportadas?

- La lista la pueden encontrar en los apuntes y las clases del curso:
  - *Clase 8 slides 17 y 22, Clase 9 slides 11, 14 y 31*
  - ***Apunte 4 págs 14 y 18, Apunte 5 págs 8 y 17***
- **Ojo!** ∑(°□° 〃) Las siguientes instrucciones **NO** están soportadas por el emulador (#Issue 61):

- `NOT A,Lit`
- `NOT A,(Dir)`
- `NOT A,(B)`
- `SHL A,Lit`

- `SHL A,(Dir)`
- `SHL A,(B)`
- `SHR A,Lit`
- `SHR A,(Dir)`
- `SHR A,(B)`

Condition Codes y Saltos Condicionales

Registro Status guarda los valores de los *condition codes* **Z**ero, **N**egative, **C**arry, o**V**erflow.

Usamos estos condition codes (o *flags*) para determinar si se realizan los saltos condicionales.

ε=ε= ε=(☞ﾟ∀ﾟ)☞

Instrucción	Operandos	Operación	Condiciones	Ejemplo de uso
CMP	A,B	A-B		
	A,(B)	A-Mem[B]		
	A,Lit	A-Lit		CMP A,0
	A,(Dir)	A-Mem[Dir]		CMP A,(label)
JMP	Dir	PC = Dir		JMP end
JEQ	Dir	PC = Dir	Z=1	JEQ label
JNE	Dir	PC = Dir	Z=0	JNE label
JGT	Dir	PC = Dir	N=0 y Z=0	JGT label
JLT	Dir	PC = Dir	N=1	JLT label
JGE	Dir	PC = Dir	N=0	JGE label
JLE	Dir	PC = Dir	Z=1 o N=1	JLE label
JCR	Dir	PC = Dir	C=1	JCR label
JOV	Dir	PC = Dir	V=1	JOV label

Subrutinas

(ʔ´ ᵕ `)ʔ⁄*✧⁺°

- Nos servirán para modularizar código de nuestro programa.
- Necesitan 3 elementos:
  - Parámetros de entrada → Pueden almacenarse en registros o memoria
  - Valor de retorno
  - Llamada a la subrutina (CALL) y retorno (RET) → No sabemos donde volver por lo que no podemos usar saltos. Guardaremos el valor del **PC+1**
- Podemos hacer uso del **stack** para guardar nuestros valores de registros, previo a la llamada de la subrutina con PUSH ← Para recuperar este valor usamos POP

Instrucción	Operandos	Operación	Condiciones	Ejemplo de uso
CALL	Dir	Mem[SP] = PC + 1, SP- -, PC = Dir		CALL func
RET		SP++		-
		PC = Mem[SP]		-
PUSH	A	Mem[SP] = A, SP- -		-
PUSH	B	Mem[SP] = B, SP- -		-
POP	A	SP++		-
		A = Mem[SP]		-
POP	B	SP++		-
		B = Mem[SP]		-