REGLAS Y DIRECTIVAS DE ENSAMBLADOR

Estructura de módulos en ensamblador

Considere lo siguiente:

```
        AREA ARMex,   CODE,   READONLY
        ;   Name this block of code ARMex
        ENTRY                       ;   Marca la primera instrucción a ejecutarse
Start   Mov     r0,   #10     ;   Inicializa (set) los parámetros
        MOV     r1,   #3
        ADD     r0,   r0,   r1 ;   r0 = r0+r1
Stop    B       stop         ;   loop infinito
        END                  ;   marca el final del archivo
```

Mientras la rutina podría parecer un poco oculta, solamente realiza una cosa: suma los números 10 y 3. El resto del código consiste en directivas para que el ensamblador y la instrucción al final, logre que el programa salga de un loop infinito. Note que existen estructura en las líneas del código, y en la forma general de la fuente de las líneas en los archivos de ensamblador, siendo:

{etiqueta} {instrucción | directiva | pseudo-instrucción} { ; comentario}

Tenga en cuenta que la etiqueta sólo puede definirse una vez en el código, y deben iniciar en el extremo izquierdo de la línea. Las instrucciones, directivas y pseudo-instrucciones deben estar precedidas de un espacio (Tab), aunque no se tenga una etiqueta en la misma línea.
El lenguaje ensamblador usado en la actualidad se llama UAL (Unified Assembler Language), entre los cambios con la versión anterior, se presentan ambos formatos para una instrucción de intercambio:

Old ARM format	UAL format
`MOV <Rd>, <Rn>, LSL shift`	`LSL <Rd>, <Rn>, shift`
`LDR{cond}SB`	`LDRSB{cond}`
`LDMFD sp!,{reglist}`	`PUSH {reglist}`

ARM Version 4T Instruction Set

ADC	ADD	AND	B	BL
BX	CDP	CMN	CMP	EOR
LDC	LDM	LDR	LDRB	LDRBT
LDRH	LDRSB	LDRSH	LDRT	MCR
MLA	MOV	MRC	MRS	MSR
MUL	MVN	ORR	RSB	RSC
SBC	SMLAL	SMULL	STC	STM
STR	STRB	STRBT	STRH	STRT
SUB	SWI[a]	SWP	SWPB	TEQ
TST	UMLAL	UMULL		

[a] The SWI instruction was deprecated in the latest version of the ARM *Architectural Reference Manual* (2007c), so while you should use the SVC instruction, you may still see this instruction in some older code.

Normalmente, los comentarios son de utilidad en proyectos muy largos, para ayudar a recordar la idea de por qué se colocó esa instrucción, a continuación, se presentan una guía de reglas básicas para escribir comentarios:

- No comente lo obvio
- Use lenguaje conciso para describir el valor que tienen los registros o cómo una función se comporta
- Comente las secciones de código donde usted considere que otro programador puede tener dificultad en comprender la razón de la instrucción. Los algoritmos complicados usualmente requieren un entendimiento más profundo en el código, y con la documentación necesaria, se pueden hacer más rápidos.
- De ser posible, evite abreviaturas
- Están prohibidos los acrónimos (abreviatura, sinónimo), pueden confundirse con instrucciones

Dentro de las herramientas Keil, las comillas en una línea, indican el inicio de un comentario, a menos que se tengan dentro de una constante de cadena, por ejemplo:

```
Abc     SETS ¨Esta es un punto y coma ;¨
```

Acá, una oración (string) es asignada a la variable abc, pero como el punto y coma se encuentra entre comillas, no existe comentario en esta línea.
En algún punto, iniciará a usar constantes en su programación, y es permitido su uso en los siguientes formatos:

- Decimal, ejemplo 123
- Hexadecimal, ejemplo 0x3F
- N_xxx (sólo Keil) donde:
    - N es base entre 2 y 9
    - Xxx es un número en esa base

Las constantes de caracter, se pueden colocar abriendo y cerrando comillas.

Definiendo un bloque de datos o códigos
Cuando se crea un código, las herramientas necesitan que se les diga cómo se les tratará o trabajará, es decir, secciones de daos, secciones de programa, bloque de coeficientes, etc. Estas secciones son indivisibles y nombradas, luego se trabajarán en la memoria adecuada para el sistema.

Frequently Used Directives

Keil Directive	CCS Directive	Uses
AREA	.sect	Defines a block of code or data
RN	.asg	Can be used to associate a register with a name
EQU	.equ	Equates a symbol to a numeric constant
ENTRY		Declares an entry point to your program
DCB, DCW, DCD	.byte, .half, .word	Allocates memory and specifies initial runtime contents
ALIGN	.align	Aligns data or code to a particular memory boundary
SPACE	.space	Reserves a zeroed block of memory of a particular size
LTORG		Assigns the starting point of a literal pool
END	.end	Designates the end of a source file