UAL Assembly language changes

Table 1 shows the main differences between UAL and the earlier ARM assembler language. The pre-UAL ARM syntax is accepted by the assembler.

Table 1: Changes from earlier ARM assembly language

Change	Pre-UAL ARM syntax	Preferred syntax
The default addressing mode for LDM and STM is IA	LDMIA, STMIA	LDM, STM
You can use the PUSH and POP mnemonics for full, descending stack operations in ARM as well as Thumb.	STMFD sp!, {*reglist*} LDMFD sp!, {*reglist*}	PUSH {*reglist*} POP {*reglist*}
You can use the LSL, LSR, ASR, ROR, and RRX instruction mnemonics for instructions with rotations and no other operation, in ARM as well as Thumb.	MOV *Rd*, *Rn*, LSL *shift* MOV *Rd*, *Rn*, LSR *shift* MOV *Rd*, *Rn*, ASR *shift* MOV *Rd*, *Rn*, ROR *shift* MOV *Rd*, *Rn*, RRX	LSL *Rd*, *Rn*, *shift* LSR *Rd*, *Rn*, *shift* ASR *Rd*, *Rn*, *shift* ROR *Rd*, *Rn*, *shift* RRX *Rd*, *Rn*
Use the *label* form for PC-relative addressing. Do not use the *offset*form in new code.	LDR *Rd*, [pc, *#offset*]	LDR *Rd*, *label*
Specify both registers for doubleword memory accesses. You must still obey rules about the register combinations you can use.	LDRD *Rd*, *addr_mode*	LDRD *Rd*, *Rd2*, *addr_mode*
{*cond*}, if used, is always the last element of all instructions.	ADD{*cond*}S LDR{*cond*}SB	ADDS{*cond*} LDRSB{*cond*}
You can use both ARM {*cond*} conditional forms and Thumb-2 ITinstructions, in both ARM and Thumb-2 code. The assembler checks for consistency between the two, and assembles the appropriate code depending on the current instruction set. If you omit the IT instructions, the assembler inserts them for you in Thumb-2 code.	ADDEQ r1, r2, r3 LDRNE r1, [r2, r3]	ITEQ E ; optional ADDEQ r1, r2, r3 LDRNE r1, [r2, r3]

In addition, some flexibility is permitted that was not permitted in previous assemblers (see Table 2).

Table 2: Relaxation of requirements

Relaxation	Permitted syntax	Preferred syntax
If the destination register is the same as the first operand,	ADD r1, r3	ADD r1, r1, r3

Relaxation	Permitted syntax	Preferred syntax
you can use a two register form of the instruction.		

You can write source code for pre-Thumb-2 Thumb processors using UAL.

If you are writing Thumb code for a pre-Thumb-2 processor, you must restrict yourself to instructions that are available on the processor. The assembler generates error messages if you attempt to use an instruction that is not available.

If you are writing Thumb code for a Thumb-2 processor, you can minimize your code size by using 16-bit instructions wherever possible.

Table 3 shows the main differences between pre-UAL Thumb assembly language and UAL. The assembler accepts the pre-UAL Thumb syntax only if it is preceded by a CODE16 directive, or if the source file is assembled with the --16 command-line option.

Table 3: Differences between pre-UAL Thumb syntax and UAL syntax

Change	Pre-UAL Thumb syntax	UAL syntax
The default addressing mode for LDM and STM is IA	LDMIA, STMIA	LDM, STM

You must use the S postfix on instructions that update the flags. This change is essential to avoid conflict with 32-bit Thumb-2 instructions.	ADD r1, r2, r3
	SUB r4, r5, #6
	MOV r0, #1
	LSR r1, r2, #1
		ADDS r1, r2, r3
		SUBS r4, r5, #6
		MOVS r0, #1
		LSRS r1, r2, #1

The preferred form for ALU instructions specifies three registers, even if the destination register is the same as the first operand.	ADD r7, r8 SUB r1, #80	ADD r7, r7, r8 SUBS r1, r1, #80

If *Rd* and *Rn* are both Lo registers, MOV *Rd*, *Rn* is disassembled as ADDS *Rd*, *Rn*, #0.	MOV *r2*, *r3*
	MOV *r8*, *r9*
	CPY *r0*, *r1*
	LSL *r2*, *r3*, #0
		ADDS *r2*, *r3*, #0
		MOV *r8*, *r9*
		MOV *r0*, *r1*
		MOVS *r2*, *r3*

NEG *Rd*, *Rm* is disassembled as RSBS *Rd*, *Rm*, #0.	NEG *Rd*, *Rm*	RSBS *Rd*, *Rm*, #0

Source:
http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0204i/Cjagjjbc.html