Appendix I – page 440

(describing BASIC development to version 1.05)

add:

BASIC V and BASIC VI continued to be developed as the hardware platforms developed. The improvements in each version are described below but the timeline is given below so that the version numbers can be tied to the various hardware platforms. Where features are described in this manual that are not present in all versions of BASIC V and BASIC VI (i.e. were added after version 1.05) then they will be so marked.

The Acorn Archimedes was launched in June 1987 and by 1992 was available in several models including the A400 series (with an ARM 3 processor), the A3000 series (with an ARM250 processor) and the portable A4 (with an ARM3 preocessor). All of these contained RISC OS 3.10 and BASIC version 1.05 (10-Apr-1992) on which issue 1 of the BBC BASIC Reference Manual was based, published in October 1992. The principal hardware developments that followed are described below – these are complicated by the fact that BASIC was developed in two forks from version 1.20, reaching version 1.36 in the RISC OS Limited fork before that line of development became defunct.

Platform	Processor	RISC OS	BASIC	
Virtual A5000	(emulated) ARM3	3.11	1.05 (10-Apr-1992)	
Risc PC	ARM610	3.50	1.06 (23-Aug-1993)	
A7000	ARM7500	3.60	1.14 (17-Nov-1994)	ARM 7500 32MHz
Risc PC	ARM710	3.60	1.14 (17-Nov-1994)	
Risc PC	StrongARM	3.70	1.16 (01-Apr-1996)	
A7000+	ARM7500FE	3.71	1.16 (01-Apr-1996)	ARM 7500FE 48MHz
Virtual Risc PC-SE	ARM 710 (emulated)	4.02	1.19 (13-Jul-1999)	
• *Kinetic Risc PC*	*StrongARM*	*4.03*	*1.20 (15-Sep-1999)*	• *Select fork*
• *Select 1i1*	*-*	*4.24*	*1.22 (21-Jun-2001)*	• *Select fork*
• *Select 1i3*	*-*	*4.29*	*1.23 (22-Feb-2002)*	• *Select fork*
• *Select 2i3* • *& Kinetic RPC*	*StrongARM*	*4.33*	*1.24 (30-Oct-2002)*	• *Select fork*
• *Select 3i1*	*-*	*4.36*	*1.27 (02-Mar-2003)*	• *Select fork*
• *Virtual RPC-Adj* • *&* • *Select 3i4*	*StrongARM or ARM710 (emulated)*	*4.39*	*1.29 (08-Jan-2004)*	• *Select fork*
• *Select 4*	*-*	*4.42*	*1.34 (06-Oct-2004)*	• *Select fork*
Iyonix	Xscale 80321	5.01	1.34 (02-Dec-2002)	80321 600MHz
Iyonix	Xscale 80321	5.03	1.35 (03-Mar-2003)	
Iyonix	Xscale 80321	5.07	1.36 (21-Jun-2004)	
Iyonix	Xscale 80321	5.10	1.37 (27-May-2005)	
Iyonix	Xscale 80321	5.14	1.39 (04-Oct-2008)	
Iyonix	Xscale 80321	5.16	1.44 (16-Jun-2009)	
Iyonix	Xscale 80321	5.18	1.48 (11-Dec-2011)	
Beagleboard	OMAP3530	5.18	1.48 (11-Dec-2011)	ARM Cortex-A8 720MHz
Beagleboard-XM	DM 3730	5.18	1.48 (11-Dec-2011)	ARM Cortex-A8 1GHz
Raspberry Pi	BCM 2835	5.19	1.53 (18-Sep-2012)	RC6, RC8 ARM1176JZF 700MHz
Iyonix	Xscale 80321	5.20	1.54 (27-Apr-2013)	
Beagleboard	OMAP3530	5.20	1.54 (27-Apr-2013)	ARM Cortex-A8 720MHz
Beagleboard-XM	DM 3730	5.20	1.54 (27-Apr-2013)	ARM Cortex-A8 1GHz
Raspberry Pi	BCM 2835	5.21	1.54 (27-Apr-2013)	RC11 ARM1176JZF 700MHz
Pandaboard ES	OMAP4460	5.21	1.54 (27-Apr-2013)	ARM Cortex-A9 1.2GHz

Development of RISC OS 3 reached 3.70/3.71 in 1996 which included StrongARM compatability, but there was little change in BASIC functionality between version 1.06 (RISC OS 3.50), 1.16 (RISC OS 3.70) and 1.19 (RISC OS 4.02). Some development took place under the 'Select' scheme (mostly, if not exclusively, confined to bug fixes) up to version 1.34 (06-Oct-2004) (RISC OS 4.42). Meanwhile development was taking place, in parallel, for the Iyonix computer, which was released with version 1.34 (02-Dec-2002) (RISC OS 5.01).

Whilst the changes at each version are described in some detail below, there are good reasons to review how developments between 1999 (RISC OS 4.02 and 4.03) and 2002 (RISC OS 4.33 and RISC OS 5.01) took place, if only to retain clarity between the Select fork and the Castle fork. Up to version 1.19, the version number of BASIC is unambiguous. After that point it is clearer to say 'requires RISC OS 5' or 'requires RISC OS 5.19 (18-Sep-2012) or later' or 'requires RISC OS 5.20 or later'. In each case these are equivalent to requiring at least version 1.34 (02-Dec-2002), 1.53 (18-Sep-2012) and 1.54 (27-Apr-2013) respectively but avoid confusion with Select developments.

The Iyonix computer contains a 'flash ROM' allowing a simple route to apply free updates and most users will have updated to at least RISC OS 5.16 which contained version 1.44 (16-Jun-2009). Other platforms (such as Beagleboard, Pandaboard and Raspberry Pi) will also be at least this level. Developments in the Select fork beyond RISC OS 4.42 (BASIC version 1.34) are not covered further.

The assembly language commands depend on the embedded processor as follows:

Processor	ARM type	Instruction set	Comments
ARM250	ARM 2	ARM v2	
ARM3	ARM 3	ARM v2	
ARM 610	ARM6	ARM v3	
ARM 7500	ARM7	ARM v3	
ARM 7500FE	ARM7	ARM v3	Floating Point
ARM 710	ARM7	ARM v3	
StrongARM SA-110	-	ARM v4	
Xscale 80321	-	ARM v5TE	
Broadcom BCM 2835	ARM 11	ARM v6	
OMAP 3530	ARM Cortex-A8	ARM v7	NEON VFP
DM 3730	ARM Cortex-A8	ARM v7	NEON VFP
OMAP 4460	ARM Cortex-A9	ARM v7	NEON VFP

see Appendix J for details

BASIC V Version 1.06 (23-Aug-1993) improvements

Summary:
The COLOUR, GCOL, MODE, POINT commands have been extended

Detail:
COLOUR r,g,b: set colour to r, g, b – calls ColourTrans_SetColour
COLOUR n,r,g,b: set palette entry for n to r, g, b physical colour (now the r,g,b values are not confined to the top four bits).

GCOL [<action>,]r,g,b: set colour to r, g, b, (calls ColourTrans_SetGCOL)..

MODE <number>|<string> - now takes a numeric or a string parameter: where the numeric parameter is greater than 255, it calls OS_ScreenMode; where it is given a string parameter, it calls *WimpMode <string>, see PRM-5 for more information about this string.
As a function MODE gives the current screen mode, numeric or a pointer to a string.

MODE has been extended to take a pointer to a mode selector or a mode string. A mode selector is passed to OS_ScreenMode, whereas a mode string is passed to *WimpMode. The result is that if a program changes

screen mode using a mode string, e.g. MODE "X800 Y600 C256 EX1 EY1", then when the program finishes that will be your desktop mode.

It should also be noted that the C/G option in the mode string is implemented by *WimpMode. Hence, MODE MODE will select a default palette mode for the given current mode depth. This means that if a grey mode is selected by MODE "X800 Y600 G256 EX1 EX1" then MODE MODE will revert back to the coloured palette.

POINT (X,Y) now supports 16bpp and 32bpp colours.

BASIC V Version 1.20 (15-Sep-1999) improvements (forks still together)

Summary:
Some bug fixes

Detail:
Messages updated to match new functionality of GCOL, COLOUR and MODE
BASIC VI (BASIC64) is now built and installed in HardDisc4 in !System.Modules

BASIC V improvements in the Select fork from 1.20 to 1.34 (06-Oct-2004)

Release: Select 1i1 – version 1.22
- *Fix for MVN conversion getting MOV r0,#&80000000 wrong.*
- *Added include of Copyright string.*

Release: Select 1i3 – version 1.23
- *Additional check for resizes outside our control.*

Release: Select 2i3 – version 1.24
- *Fix for mistabbing in BASIC VI.*
- *Updated SOUND to use Sound_ControlPacked.*

Release: Select 3i1 – version 1.27
- *Removed BASICTrans.*
- *Fix for failed resource export.*
- *Updated to use new-style Makefiles.*
- *Removed redundant export of header file.*
- *Fix for HELP resources not being installed.*

Release: Select 3i4 – version 1.29
- *Removed rogue debug code.*
- *Updated to build with ObjAsm.*

Release: Select 4 Preview – version 1.34
- *Fix for LD/ST 'T' instructions being misassembled.*
- *Updated module to 32bit.*
- *Rationalised BASIC64 to share more code with BASIC.*
- *Updated SYS return of flags to return PSR as expected in 32bit builds.*
- *Finished making 32bit safe (high memory)*
- *Fix for error pointers returned from called code in high memory.*

BASIC V Version 1.24 (16-Mar-2001) improvements (in the ROOL/Castle fork from 1.19)

Summary:
Some bug fixes and a few new features.
Detail:
* ALIGN will now force any unused bytes to zero when assembling at

both P% and O%. This is better than leaving possibly uninitialised memory behind (makes diffs very difficult).

* The assembler now correctly recognises the new 'LR' special variable when the L is lower-case.

* The assembler also has the 'SP' (13) special variable defined.

* TRACE PROC (and the identical TRACE FN) now flushes the VCACHE when it is encountered.

* TRACE ENDPROC has been implemented to allow the output of 'ENDPROC' and 'ENDFN' in the trace output whenever a procedure or function is exited. This complements the TRACE PROC/FN functionality.

* QUIT <expression> has been added to allow a return code to be passed back to the called when BASIC exits. This is the /only/ sensible way for a BASIC program to set Sys$ReturnCode on exit (others get trashed when BASIC removes it's environment handlers).

BASIC V Version 1.26 (20-Apr-2001) improvements

Summary:
* New forms of MODE, COLOUR and GCOL added.
* VDU can now be used as a function to read VDU variables.
* Will surrender application memory for MODE changes etc.
Detail:
* New syntax for MODE: MODE <width>,<height>,<bpp>[,<framerate>]
  Uses OS_ScreenMode 0, rather than *WimpMode. bpp may be 1,2,4,6,8,16 or 32. 8 selects a full palette 256-colour mode, 6 an old-style one.
* COLOUR and GCOL both allow colour numbers rather than old-style colours. Also, background colours can be set with R,G,B forms.
  COLOUR [OF f] [ON b]
  COLOUR [[OF] r,g,b] [ON r,g,b]
  GCOL [OF [<action>],f] [ON [<action>],b]
  GCOL [[OF] [<action>,]r,g,b] [ON [<action>,]r,g,b]

  For COLOUR R,G,B, the OF is unnecessary, but provided for uniformity.
  For GCOL R,G,B, OF tightens up the usage of <action> - without it <action>is passed in to both R3 and R4 of ColourTrans_SetGCOL for backwards compatibility (some may have used GCOL 128,r,g,b to set the background colour - although this ends up setting reserved flags in the ColourTrans_SetGCOL call).
* Used as a function VDU n returns the value of the specified VDU variable.
* BASIC will now surrender application space if possible at certain moments: during SYS calls (as long as no string parameters are passed in), MODE changes, OSCLI and * commands. Also, it will not refuse requests to grow application space (although it will not expand into the extra space).

  Application space is only surrendered if no library is INSTALLed, HIMEM is set to the top of BASIC's memory, and BASIC's memory extends to the top of the application slot.

  This permits easy MODE changes etc outside the desktop. Note the effect that now outside the desktop, with screen memory at <300K and no free pool, MODE 28 will work, while VDU 22,28 will not. This

restores some long lost BBC MOS behaviour.

BASIC V Version 1.30 (23-Apr-2002) improvements

Summary:
* Performance tweaks.
* Comments added.
* Added DIM LOCAL.
* Fix to *BASIC command.

Detail:
* All of the 408 instances of pushes and pops of a single register (i.e. "LDMIA SP!,{PC}") have been replaced with an LDR/STR equivalent, which provides a small performance boost on StrongARM-like processors.
* Some obscure bits of the source have had comments added.
* The new DIM LOCAL statement has been added:

Syntax: DIM <numeric variable> LOCAL <numeric expression>

DIM LOCAL allocates space from the BASIC stack rather than from the heap. This means that the space allocated is automatically freed on exit from the PROC or FN in which it was claimed.

As with all LOCAL statements, it must appear in a PROC or FN definition and cannot appear inside a structure which uses the stack (i.e. NEXT) or after a LOCAL ERROR.

Much like the manner in which "DIM foo% -1" returns the value of FSA without allocating space, "DIM foo% LOCAL -1" returns the value of SP at the time of the request, without allocating any space from the stack. This can be useful in computing the free space between the top of the BASIC heap and the bottom of the stack. Of course, the stack is used by BASIC while interpreting the program, so the contents of memory below foo% should always be treated as undefined.

One caveat of using DIM LOCAL is that the stack cannot be moved while there are DIM LOCALs defined. END= will return an error and any Service_Memory calls will be claimed for the duration of the DIM LOCAL.

* *BASIC will now return an error if it fails for any reason. This is slightly more polite than the SVC mode infinate loop it used to go into.

BASIC V Version 1.34 (02-Dec-2002) improvements

Summary:
Fixed bug spotted on csa.programmer in BASIC CASE statements.

Detail:
The expression evaluation in the BASIC WHEN statement had an omission where it could skip past a syntax error without generating any error.

For example, "WHEN (R%>>25) AND 1)=1 : do something" would not return an error because the expression is complete and evaluated upto the "AND 1". Once this is found to not match the condition of the CASE, BASIC searches forwards for the next ",", "WHEN", "OTHERWISE" or "ENDCASE", skipping over the ")=1" part.

I've changed this so that, after the expression is evaluated and found

to not match, BASIC already checks to see if the next char is a ",". If
not, we check for a ":" or CR before scanning forwards. Any other char
is a syntax error.

Fixed a couple of >64M address problems.

Bugfix: the TITLE routine was stacking and unstacking R13. This is defined
as UNPREDICTABLE in the ARM ARM, and while it seems to have been harmless on
older ARMs, the XScale corrupts R13 - in this case, it was always loaded with
the value &80F0. This meant that you couldn't load untokenised BASIC programs
via the *BASIC command because TOP was already above the bottom of the stack!
Also optimised a non-aligned data copy routine a little bit.

BASIC V Version 1.36 (21-Jun-2004) improvements

Summary:
Some bug fixes
A new variant of the COLOUR command added

Detail:
* In BASIC64, A() = B() / C now works - this specific case failed due to
  a stack imbalance.
* Keywords that take tokenised line numbers no longer cause number
  tokenisation if they appear on the right. This fixes BPUT#TRACE,32
* STRT/LDRT now enforce post-indexing. LDRT R0,[R1] generates correct code,
  and LDRT R0,[R1,#0] will be faulted.

COLOUR – the new four-parameter form of the COLOUR command allows setting of the supremacy bits of palette entries.

BASIC V Version 1.37 (27-May-2005) improvements

Previous fix for BPUT#TRACE,32 broke tokenisation of IF F% THEN 30. Both now
handled correctly.

BASIC V Version 1.39 (04-Oct-2008) improvements

Merged in changes from Castle
Detail:
  Introduced error out of range check for B and BL instructions
  in Basic Assembler   file Assembler.s
Admin:
  Not tested. Version incremented by 2 to match Castle's version.

BASIC V Version 1.42 (10-May-2009) improvements

Fix BASIC to handle ARMv6/ARMv7 unaligned loads

Detail:
  s/Command - LOADFILETOKENCOPY now uses compile-time architecture flags to decide how
  to handle the potential unaligned load while copying data
  s/ModHead - Include Hdr:CPU.Arch to get architecture flags

BASIC V Version 1.43 (17-May-2009) improvements

Unaligned load optimisations

Detail:

  * Uses global LDW macro instead of locally-defined LOAD macro (or longhand
    implementations)
  * Some other cases of unaligned loads for v6+ added in places that didn't
    lend themselves to the LDW macro
  * 5-byte (FP) load operation now macroised - the macro skips the unnecessary
    BIC when assembling only for pre-v6 CPUs, or uses unaligned loads for
    v6+ CPUs (unless NoUnaligned global flag is set)

BASIC V Version 1.44 (16-Jun-2009) improvements

Speed improvements on ARMv6 or later

Detail:
  s.Assembler line 1861: incorrect LDR of 1-byte variable BYTESM. Only bit 2
    of value tested, so no ill effects other than inefficiency on ARMv6.
  s.Stmt2 and s.fp: optimised stores of 5-byte floating-point values to use
    unaligned STR on ARMv6 or later.
  Added ENTRY directive to permit building of GPA debug listing

BASIC V Version 1.48 (11-Dec-2011) improvements

Make TEXTLOAD do RENUMBER 10,1 when renumering is needed
Effectively raises the longest text program you can load to ~64k lines

BASIC V Version 1.49 (19-Dec-2011) improvements

Add TBA Software's VFP/NEON assembler
Detail:
  This is version 0.06 of TBA's VFP/NEON BASIC assembler, with the following modifications:
  * VFPASM makefile option can be used to toggle the assembler on/off. Currently only used by the Tungsten ROM, due to lack of ROM space.
  * EnableVFPDebug switch added to turn on/off debugging code
  * VFP_Error macro fixed (VFPError symbol wasn't defined in 0.06 sources; now it just calls MSGERR directly)
  * Debug messages tweaked to make them a bit more useful
  * Single-register LDM/STM swapped for LDR/STR to avoid objasm 4 performance warnings
  * Fixed a bug in VFPop_imm_bits that caused parsing of immediate constants to fail in ROM builds, and potentially fail or misbehave in RAM builds. A sequence of conditional instructions incorrectly had the 'S' flag set on each instruction, which could cause the execution to stop halfway through.
  * VFPLib BASIC file moved from Tests.VFP folder into all-new VFPLib folder
  * Makefile modified and VFPLib.GenData script added, in order to allow s.VFPData to be built automatically as part of the build process instead of relying on a prebuilt version in CVS
  File changes:
  - Doc/VFPdoc - VFP assembler implementation notes & usage notes
  - Tests/VFP/!Setup,feb, Tests/VFP/TestVFP,ffb, Tests/VFP/VFPLibAsm,ffb, Tests/VFP/VFPLibTest,ffb - VFP test programs
  - VFPLib/GenData,ffb, VFPLib/VFPLib,ffb - Common BASIC files used to generate the pattern tables used by the assembler, and used by the test programs
  - s/VFP, hdr/VFPMacros - Main VFP/NEON assembler source code
  - Makefile - Modified to allow automatic generation of s.VFPData file
  - s/Assembler - Modified to call VFP assembler for instructions starting with 'V'
  - s/ModHead - Replaced ADRL with a sequence of ADD instructions due to module now being over 64K in size

BASIC V Version 1.50 (07-Mar-2012) improvements

VFP/NEON assembler fixes and improvements
Detail:
  * Reworked to make ROM builds reference VFP/NEON assembler lookup tables held in
  ResourceFS instead of using local copies held in the module. Saves ~32K of ROM space

due to reduced data duplication between BASIC & BASIC64.
* Replace OPT flag magic numbers with symbols
* Added support for VFPv4/ASIMDv2 instructions (VFMA/VFMS, VFNMA/VFNMS)
* Updated DCFD/EQUFD to add support for .VFP and .FPA suffixes, to indicate whether VFP or FPA word order should be used
* Added DCFH/EQUFH for storing half precision floats (in advanced half precision format)
* Rework MSG routine to preserve R4-R7, for passing to BASICTrans as message parameters
* Added a few new error messages (118-123) for the VFP/NEON assembler
* Enable OWNERRORS option for standalone builds, and set DO32BIT to 0 for 26bit builds
* Fixed VLDM/VSTM style register lists not working correctly when using commas to list the registers
* Improved handling of VFP/NEON 8 bit immediate constants:
  - No longer possible to use #I64.<n>, #F32.<n>, #F64.<n> notation to directly specify the 8-bit encoded value for I64/F32/F64 constants, nor to force one instruction to attempt to use a constant in a format that doesn't match the .<dt> instruction suffix
  - Instead, .I64 constants can either be specified as a number (which will be converted to an integer and then zero-extended to 64 bits) or a string (which will be parsed by the 64bit version of OS_ReadUnsigned)
  - .F32 and .F64 constants should now be specified as floating point numbers which will then be correctly converted to the 8 bit encoding
  - .F32 and .F64 constants which can't be converted will result in a error message indicating the closest possible floating point number that can be encoded (but not necessarily the closest number that can be encoded; e.g. when assembling NEON instructions the assembler can automatically make use of .I32 encoding formats for some numbers)
* Fixed instructions that have an 'imn' field (e.g. VSHR immediate) being incorrectly assembled
* VFPLib.GenData & VFPLib.VFPLib now stored as text instead of tokenised BASIC
Files changed:
* Makefile, VFPLib.GenData, VFPLib.VFPLib, hdr.VFPMacros, hdr.Workspace, s.Assembler, s.Basic, s.ErrorMsgs, s.Factor, s.ModHead, s.VFP

BASIC V Version 1.51 (25-Mar-2012) improvements

Fix min/max limits of NEON immediate shift instructions
Detail:
VFPLib/VFPLib - Added "g" and "h" immediate constant types. Updated descriptions of existing types. Fixed VQSHL{U}, VSHL, VSHLL and VSLI to use the correct immediate types.
s/VFP - Added support for the new immediate constant types. Fixed VFPop_imm6 masking the immediate value against (1<<bits)-1, causing the wrong constant to be encoded when imm==size

BASIC V Version 1.52 (09-Jul-2012) improvements

Protect against reading null pointers when using the output of a SWI to fill a string
Detail:
s/Stmt2 - Code around SYS1STRING now treats null pointers returned by SWIs as being null strings. This protects against a crash if the user expects a SWI to return a string but it decides not to (e.g. due to an error)

BASIC V Version 1.53 (18-Sep-2012) improvements

Add an attempt to load ArmBE when EDIT keyword is used
It may fail if not available, but so did 'EDIT'. This way if it is available at least

the 'EDIT' command enters the editor.
Note: this replaces the recommendation to load from ADFS::0.$.Modules.BasicEdit in the chapter "Editing BASIC files under RISC OS 2" in the BBC BASIC reference manual (!).

BASIC V Version 1.54 (26-Apr-2013) improvements

Changed Basic module so it correctly reports ERL after an external abort.
Detail:
There was a significant problem with Basic when external code is called using CALL, USR, SYS, or a *command (either directly or using OSCLI), and that code Aborts with Data Transfer, Undefined Instruction or Instruction Fetch. The Error Line (ERL) then given by Basic was often the last line of the program, rather than the Basic line containing the CALL, USR, SYS or *command.

The problem was caused when the Basic LINE pointer in R12 (which points to the current line) was corrupted, and that is used by Basic to find the program line number (ERL) after an error. If R12 is outside the program, ERL was set to zero or the last line of the program, which was useless for problem diagnosis.

After an Abort in a SYS, or in a module *Command (issued direct or via OSCLI), R12 is always corrupted. After an Abort in code executed from a CALL or USR then R12 is corrupted only if the called code changed it.

In addition, if r13_usr was corrupted, the Basic error handler was reset to the default, causing the program to end abruptly without any line number at all.

Appendix J

Add to end of this Appendix:

The assembly language commands depend on the embedded processor as follows:

Processor	ARM type	Instruction set	Comments
ARM250	ARM 2	ARM v2	
ARM3	ARM 3	ARM v2	
ARM 610	ARM6	ARM v3	
ARM 7500	ARM7	ARM v3	
ARM 7500FE	ARM7	ARM v3	Floating Point
ARM 710	ARM7	ARM v3	
StrongARM SA-110	-	ARM v4	
Xscale 80321	-	ARM v5TE	
Broadcom BCM 2835	ARM 11	ARM v6	
OMAP 3530	ARM Cortex-A8	ARM v7	NEON VFP
DM 3730	ARM Cortex-A8	ARM v7	NEON VFP
OMAP 4460	ARM Cortex-A9	ARM v7	NEON VFP

Changes from ARM v2 to v3

No major change to assembler. Note that the ARM 7500FE processor had floating point instructions implemented in hardware and a software floating point emulator was available for other processors. Further developments for co-processors are not discussed further except under ARM v7 for the NEON VFP.

Changes from ARM v3 to v4

The processor model changed from a ‘von Neumann’ model where data and instructions held in memory were cached in the same way to a model where separate and data caches were maintained. No major change to assembler, although Xscal had instructions MIA, MIAPH, MIAxy, MRA and MAR added.

Changes from ARM v4 to v5TE

Saturating add [Q{D}ADD Rd,Rm,Rn] added
Saturating subtract [Q{D}SUB Rd,Rm,Rn] added
Signed multiply long extended to add SMULxy, SMULWy, SMULAxy, SMULAWy and SMLALxy added
Thumb2 instructions BFC, BFI, SBFX, UBFX, RBIT, IT, BX, CBZ, TBB and TBH added
Branch with link and exchange instruction BLX added
Load/store double word instructions LDD and STD added
Preload data instruction PLD added

Changes from ARM v5TE to v6

Halfword- and byte-wise addition and subtraction; halfword-wise exchange-add-subtract and exchange-subtract-add; unsigned sum of absolute differences and accumulate instructions added:
<prefix><instruction> Rd,Rm,Rs
  where <prefix> is ?? and <instruction> is ADD16, SUB16, ADD8, SUB8, ASX, or SAX
USAD8 Rd,Rm,Rs
USADA8 Rd,Rs,Rm,Rn
Saturate instructions SSAT, SSAT16, USAT and USAT16 added
Multiply extended to add SMUAD, SMLAD, SMLALD, SMUSD, SMLSD, SMLSLD, SMMUL, SMMLA and SMMLS
Pack instructions PKHBT and PKHTB added
Signed extend instructions SXTH, SXTB16, SXTB, SXTAH, SXTAB16 and SXTAB added
Unsigned extend instructions UXTH, UXTB16, UXTB, UXTAH, UXTAB16 and UXTAB added
Reverse instructions REV, REV16 and REVSH added
Select instruction SEL added
Processor state instructions CPSID, CPSIE, CPS and SETEND added
Load exlusive instruction LDREX added
Store exclusive instruction STREX added

Changes from ARM v6 to v7

Preload instruction instruction PLI added
NEON VFP instructions (e.g. VFPop) added