Application Note 04

Programmer's Model for Big-Endian ARM

ARM POWERED

ARM™

ARM
Advanced RISC Machines

Proprietary Notice

ARM, the ARM Powered logo, BlackICE and ICEbreaker are trademarks of Advanced RISC Machines Ltd.

Neither the whole nor any part of the information contained in, or the product described in, this application note may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this application note is subject to continuous developments and improvements. All particulars of the product and its use contained in this application note are given by ARM in good faith. However, all warranties implied or expressed, including but not limited to implied warranties or merchantability, or fitness for purpose, are excluded.

This application note is intended only to assist the reader in the use of the product. ARM Ltd shall not be liable for any loss or damage arising from the use of any information in this application note, or any error or omission in such information, or any incorrect use of the product.

Change Log

Issue	Date	By	Change
A	Jul. 93	AT	Document tranfer to Frame
B	Nov. 94	AW	Update of all addresses
C	Dec. 94	AW	Reformat into new style

Application Note 04

ARM DAI 0004C

1   Introduction

The earlier ARM processors (ARM2, ARM3, ARM2aS) use a little-endian architecture. Current generation ARM processors (from ARM6 onwards) have the option to operate in either little- or big-endian mode. These terms refer to the way in which multi-byte quantities, such as 32-bit words, are stored in a byte addressed memory. In a little-endian architecture, the least significant byte of the quantity is stored at the lowest memory address in the range of addresses used to store the quantity. The reverse is true in a big-endian architecture where the most significant byte is stored at the lowest address.

This document describes the behaviour of a big-endian ARM.

2   Connection to memory

A big-endian ARM architecture should be wired as follows:

Byte 0 of the memory connected to D[31:24]

Byte 1 of the memory connected to D[23:16]

Byte 2 of the memory connected to D[15:8]

Byte 3 of the memory connected to D[7:0]

Note: *The processor may be configured to generate a fault if a non word-aligned address is used during a data transfer operation.*

3   Single data transfer (LDR, STR)

A byte load (LDRB) expects the data on D[31:24] if the supplied address is on a word boundary, on D[23:16] if it is a word address plus one byte, and so on. The selected byte is placed in the bottom 8 bits of the destination register, and the remaining bits of the register are filled with zeros.

A byte store (STRB) repeats the bottom 8 bits of the source register four times across the data bus. If the byte store is word aligned, the external memory system should write D[31:24] into byte 0 of the word in memory; if the address is a word address plus one byte, D[23:16] should be written into byte 1 of the word in memory, and so on.

A word load (LDR) should generate a word aligned address. An address offset from a word boundary by 0 or 2 bytes (half-word alignment) will cause the data to be rotated into the register so that the addressed byte occupies bits 31 to 24. (An address offset from a word boundary by 1 or 3 bytes will cause the data to be rotated into the register so that the addressed byte occupies bits 15 to 8).

A word store (STR) should generate a word aligned address. The data written to memory are always presented exactly as they appear in the register (i.e. Bit 31 of the register appears on D[31]).

Programmer's Model for Big-Endian ARM

3.1   Store Operations

Initial conditions

Registers: R0=&76543210

R1=&00000001

R2=&00000002

R3=&00000003

R4=&00000004

R10=&00001000

R11=&00001004

Byte Address	1000	1001	1002	1003	1004	1005	1006	1007
Data	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.

Code segment executed:
```
STR  R0,[R10]
STRB R1,[R11,#0]
STRB R2,[R11,#1]
STRB R3,[R11,#2]
STRB R4,[R11,#3]
```

End conditions:

Registers: R0=&76543210

R1=&00000001

R2=&00000002

R3=&00000003

R4=&00000004

. . .

R10=&00001000

R11=&00001004

Byte Address	1000	1001	1002	1003	1004	1005	1006	1007
Data	76	54	32	10	01	02	03	04

3.2 LOAD OPERATIONS

Initial conditions:

Registers: R0=&FFFFFFFF

R1=&FFFFFFFF

R2=&FFFFFFFF

R3=&FFFFFFFF

R4=&FFFFFFFF

R5=&FFFFFFFF

R6=&FFFFFFFF

R7=&FFFFFFFF

. . .

R10=&00001000

Byte Address	1000	1001	1002	1003
Data	AA	BB	CC	DD

Code segment executed:

```
LDR  R0,[R10,#0]  ;word aligned load

LDR  R1,[R10,#1]
LDR  R2,[R10,#2]  ;half-word aligned load
LDR  R3,[R10,#3]

LDRB R4,[R10,#0]
LDRB R5,[R10,#1]
LDRB R2,[R10,#2]
LDRB R7,[R10,#3]
```

End conditions:

Registers: R0=&AABBCCDD

R1=&DDAABBCC

R2=&CCDDAABB

R3=&BBCCDDAA

R4=&000000AA

R5=&000000BB

R6=&000000CC

R7=&000000DD

. . .

R10=&00001000

Byte Address	1000	1001	1002	1003
Data	AA	BB	CC	DD

3.3   Block data transfer (LDM, STM)

These instructions only ever transfer word quantities, and the byte alignment of the base address has no effect on the data transferred. However, for future compatibility the address used should always be word aligned.

3.4   Single data swap (SWP)

This instruction is a merged LDR+STR operation, and so operates in the same way:-

A byte swap (SWPB) expects the read data on D[31:24] if the supplied address is on a word boundary, on D[23:16] if it is a word address plus one byte, and so on. The selected byte is placed in the bottom 8 bits of the destination register, and the remaining bits of the register are filled with zeros. The byte to be written is repeated four times across the data bus. The external memory system should activate the appropriate byte subsystem to store the data (see description of the single byte write operation).

A word swap (SWP) should generate a word aligned address. An address offset from a word boundary by 0 or 2 bytes (half-word alignment) will cause the data read from memory to be rotated into the register so that the addressed byte occupies bits 31 to 24.)An address offset from a word boundary by 1 or 3 bytes will cause the data read to be rotated into the register so that the addressed byte occupies bits 15 to 8). The data written to memory are always presented exactly as they appear in the register (i.e Bit 31 of the register appears on D[31]).

3.5   Coprocessor data transfers (LDC, STC)

These instructions only ever transfer word quantities, and the byte alignment of the base address has no effect on the data transferred. However, for future compatibility the address used should always be word aligned.

ARM
POWERED
™