Single Data Transfer (LDR, STR)
The instruction is only executed if the condition is true. The various conditions are defined in Table 6. The instruction encoding is shown in Figure 23.
The single data transfer instructions are used to load or store single bytes or words of data. The memory address used in the transfer is calculated by adding an offset to or subtracting an offset from a base register.
The result of this calculation may be written back into the base register if auto-indexing is required.

Figure 23. Single Data Transfer Instructions

Offsets and auto-indexing
The offset from the base may be either a 12 bit unsigned binary immediate value in the instruction, or a second register (possibly shifted in some way). The offset may be added to (U=1) or subtracted from (U=0) the base register Rn. The offset modification may be performed either before (pre-indexed, P=1) or after (post-indexed, P=0) the base is used as the transfer address.

The W bit gives optional auto increment and decrement addressing modes. The modified base value may be written back into the base (W=1), or the old base value may be kept (W=0). In the case of post-indexed addressing, the write back bit is redundant and is always set to zero, since the old base value can be retained by setting the offset to zero. Therefore post-indexed data transfers always write back the modified base. The only use of the W bit in a post-
Indexed data transfer is in privileged mode code, where setting the W bit forces non-privileged mode for the transfer, allowing the operating system to generate a user address in a system where the memory management hardware makes suitable use of this hardware.

**Shifted register offset**
The 8 shift control bits are described in the data processing instructions section. However, the register specified shift amounts are not available in this instruction class. See Shifts.

**Bytes and words**
This instruction class may be used to transfer a byte (B=1) or a word (B=0) between an ARM7TDI register and memory.

The action of LDR(B) and STR(B) instructions is influenced by the **BIGEND** control signal. The two possible configurations are described below.

**Figure 24. Little Endian Offset Addressing**

A word store (STR) should generate a word aligned address. The word presented to the data bus is not affected if the address is not word aligned. That is, bit 31 of the register being stored always appears on data bus output 31.

**Big endian configuration**
A byte load (LDRB) expects the data on data bus inputs 7 through 0 if the supplied address is on a word boundary, on data bus inputs 15 through 8 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. Please see Figure 5.

A byte store (STRB) repeats the bottom 8 bits of the source register four times across data bus outputs 31 through 0. The external memory system should activate the appropriate byte subsystem to store the data.

A word load (LDR) will normally use a word aligned address. However, an address offset from a word boundary will cause the data to be rotated into the register so that the addressed byte occupies bits 0 to 7. This means that half-words accessed at offsets 0 and 2 from the word boundary will be correctly loaded into bits 0 through 15 of the register. Two shift operations are then required to clear or to sign extend the upper 16 bits. This is illustrated in Figure 24.
will cause the data to be rotated into the register so that the addressed byte occupies bits 31 through 24. This means that half-words accessed at these offsets will be correctly loaded into bits 16 through 31 of the register. A shift operation is then required to move (and optionally sign extend) the data into the bottom 16 bits. An address offset of 1 or 3 from a word boundary will cause the data to be rotated into the register so that the addressed byte occupies bits 15 through 8.

A word store (STR) should generate a word aligned address. The word presented to the data bus is not affected if the address is not word aligned. That is, bit 31 of the register being stored always appears on data bus output 31.

Use of R15

Write-back must not be specified if R15 is specified as the base register (Rn). When using R15 as the base register you must remember it contains an address 8 bytes on from the address of the current instruction.

R15 must not be specified as the register offset (Rm).

When R15 is the source register (Rd) of a register store (STR) instruction, the stored value will be address of the instruction plus 12.

Restriction on the use of base register

When configured for late aborts, the following example code is difficult to unwind as the base register, Rn, gets updated before the abort handler starts. Sometimes it may be impossible to calculate the initial value.

After an abort, the following example code is difficult to unwind as the base register, Rn, gets updated before the abort handler starts. Sometimes it may be impossible to calculate the initial value.

Example:

LDRR0, [R1], R1

Therefore a post-indexed LDR or STR where Rm is the same register as Rn should not be used.

Data aborts

A transfer to or from a legal address may cause problems for a memory management system. For instance, in a system which uses virtual memory the required data may be absent from main memory. The memory manager can signal a problem by taking the processor ABORT input HIGH whereupon the Data Abort trap will be taken. It is up to the system software to resolve the cause of the problem, then the instruction can be restarted and the original program continued.

Instruction cycle times

Normal LDR instructions take 1S + 1N + 1I and LDR PC take 2S + 2N +1I incremental cycles, where S,N and I are as defined in Cycle Types.

STR instructions take 2N incremental cycles to execute.
**Assemble syntax**

\[
\text{<LDR|STR>{cond}(B|T) Rd,<Address>}
\]

where:

- **LDR** load from memory into a register
- **STR** store from a register into memory
- **{cond}** two-character condition mnemonic. See Table 6.
- **{B}** if B is present then byte transfer, otherwise word transfer
- **{T}** if T is present the W bit will be set in a post-indexed instruction, forcing non-privileged mode for the transfer cycle. T is not allowed when a pre-indexed addressing mode is specified or implied.
- **Rd** is an expression evaluating to a valid register number.
- **Rn and Rm** are expressions evaluating to a register number. If Rn is R15 then the assembler will subtract 8 from the offset value to allow for ARM7TDMI pipelining. In this case base write-back should not be specified.

**<Address>** can be:

1. An expression which generates an address:
   \[
   \text{<expression>}
   \]
   The assembler will attempt to generate an instruction using the PC as a base and a corrected immediate offset to address the location given by evaluating the expression. This will be a PC relative, pre-indexed address. If the address is out of range, an error will be generated.

2. A pre-indexed addressing specification:
   \[
   \begin{align*}
   &\text{[Rn]} & \text{offset of zero} \\
   &\text{[Rn,<#expression>]} & \text{offset of <expression> bytes} \\
   &\text{[Rn,(+/-)Rm{,<shift>}{!}]} & \text{offset of +/- contents of index register, shifted by <shift>} \\
   \end{align*}
   \]

3. A post-indexed addressing specification:
   \[
   \begin{align*}
   &\text{[Rn],<#expression>} & \text{offset of <expression> bytes} \\
   &\text{[Rn],(+/-)Rm{,<shift>}} & \text{offset of +/- contents of index register, shifted as by <shift>} \\
   \end{align*}
   \]

- **<shift>** general shift operation (see data processing instructions) but you cannot specify the shift amount by a register.
- **{!}** writes back the base register (set the W bit) if! is present.

**Examples**

\[
\begin{align*}
\text{STR R1,[R2,R4]!} & \quad ; \text{Store R1 at R2+R4 (both of which are registers) and write back address to R2.} \\
\text{STR R1,[R2],R4} & \quad ; \text{Store R1 at R2 and write back R2+R4 to R2.} \\
\text{LDR R1,[R2,#16]} & \quad ; \text{Load R1 from contents of R2+16, but don’t write back.} \\
\text{LDR R1,[R2,R3,LSL#2]} & \quad ; \text{Load R1 from contents of R2+R3*4.} \\
\text{LDREQR1,[R6,#5]} & \quad ; \text{Conditionally load byte at R6+5 into R1 bits 0 to 7, filling bits 8 to 31 with zeros.} \\
\text{STR R1,PLACE} & \quad ; \text{Generate PC relative offset to address PLACE.}
\end{align*}
\]