This section → Data Transfer Instructions

↳ single register transfer instructions

↳ Multiple register transfer instructions

↳ memory mapped I/O in ARM

Data Transfer Instructions

- ARM instruction set supports three types of data transfers.
  a) Single register loads and stores
     ↳ Flexible, supports byte, half-word and word transfers.
  b) Multiple register loads and stores
     ↳ less flexible, multiple words, higher transfer rate
  c) Single-transfer memory swap.
     ↳ mainly for system use.

- All ARM data transfer instructions are register indirect messaging.
  ↳ before any data transfer, some register must be initialized with a memory address.

```
ADRL    r1, Table    ; r1 = memory address of Table.
```

- Examples

```
LDR     r0, [r1]     ; r0 = mem[r1]
STR     r0, [r1]     ; mem[r1] = r0
```

Single register loads and store

- The simplest form uses register indirect without any offset

```
LDR     r0,[r1]      ; r0 = mem[r1]
```

```
STR     r0, [r1]        ; mem[r1] = r0
```

- An alternative form uses register indirect with offset (limited to 4 Kbytes)

```
LDR     r0, [r1,#4]     ; r0 = mem[r1+4]
STR     r0, [r1, #12]   ; mem[r1+4] = r0
```

- We can use auto-indexing in addition

```
LDR     r0, [r1, #4]!   ; r0 = mem[r1+4], r1=r1+4
STR     r0, [r1, #12]!  ; mem[r1+12] = r0, r1=r1+4
```

- We can use post indexing

```
LDR     r0, [r1], #4    ; r0= mem[r1],  r1= r1+4
STR     r0, [r1], #12   ; mem[r1] = r0,  r1 = r1+12
```

- We can specify a byte or half word to transferred.

```
LDRB    r0, [r1]        ; r0 = mem8[r1]
STRB    r0, [r1]        ; mem8[r1] = r0
LDRSH   r0, [r1]        ; r0 = mem16[r1]
STRSH   r0, [r1]        ; mem16[r1] = r0
```

Multiple register loads and stores

- ARM supports instructions that transfer between several registers and memory:

Example

```
LDMIA   r1, {r3, r5, r6}  ; r3 = mem[r1]
                          ; r5 = mem[r1+4]
```

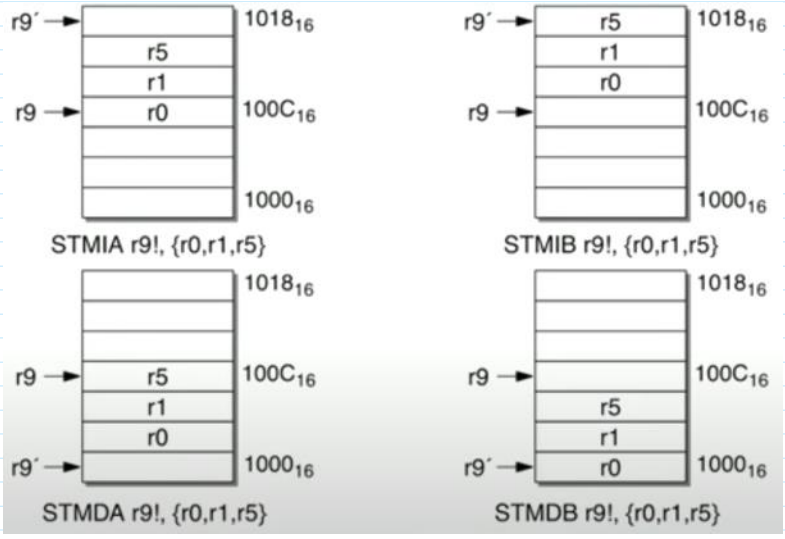
; r6 = mem[r1+8]

- For LDMIB, the addresses will be r1+4, r1+8 and r1+12
- The list of destination registers may contain any or all of r0 to r15

Block copy addressing

- supported with addresses that can increment (I) or decrement (D) before (B) or after (A) each transfer

Examples of addressing modes in multiple-register transfer:



- Point to note: → ARM does not support any hardware stack
  → Software stack can be implemented using LDM and STM

Example Copy a block of memory (128 bytes aligned)

→ r9: address of the source
→ r10: address of the destination
→ r11: end address of the source

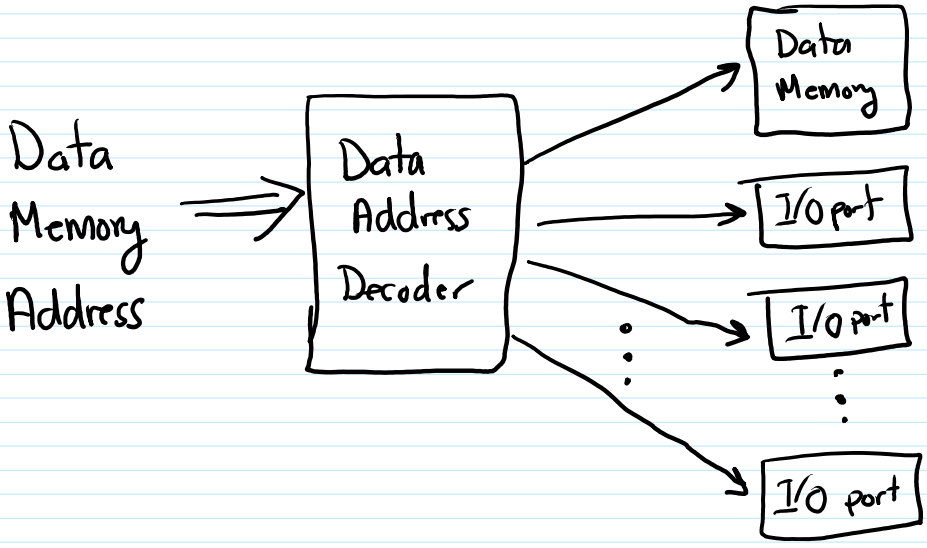
```
Loop:  LDMIA  r9!, {r0-r7}
       STMIA  r10! {r0-r7}
       CMP    r9  r11
```

STMIA r10!, {r0-r4}

CMP r9, r11 ⇝ not relevant

BNE Loop

Memory Mapped I/O in ARM



- No separate instructions for input/output
- The I/O ports are treated as data memory locations
  - ↳ each with a unique (memory) address
- Data input is done using the LDR instruction
- Data output is done with the STR instruction