MANCHEstER 1824

The University of Manchester

The ARM instruction set

- Outline:
  - privileged modes and exceptions
  - instruction set details
  - system code example

☞ hands-on: system software - SWI handler

MANCHEstER 1824

The University of Manchester

The ARM instruction set

- Outline:
  - ➜ **privileged modes and exceptions**
  - instruction set details
  - system code example

☞ hands-on: system software - SWI handler

MANCHEstER 1824

The University of Manchester

Privileged modes and exceptions

- ARM has privileged operating modes:
  - **SVC** (supervisor) mode for software interrupts
  - **IRQ** mode for (normal) interrupts
  - **FIQ** mode for fast interrupts
  - **Abort** mode for handling memory faults
  - **Undef** mode for undefined instruction traps
  - **System** mode for privileged operating system tasks

MANCHEstER 1824

The University of Manchester

Memory faults

- ARM has full support for memory faults.
  Accesses may fail because of:
  - virtual memory page faults
  - memory protection violations
  - soft memory errors
  - **Prefetch aborts** are faults on instruction fetches
  - **Data aborts** are faults on data transfers
    - both are recoverable (with a little work)
    - details vary somewhat between different ARM cores

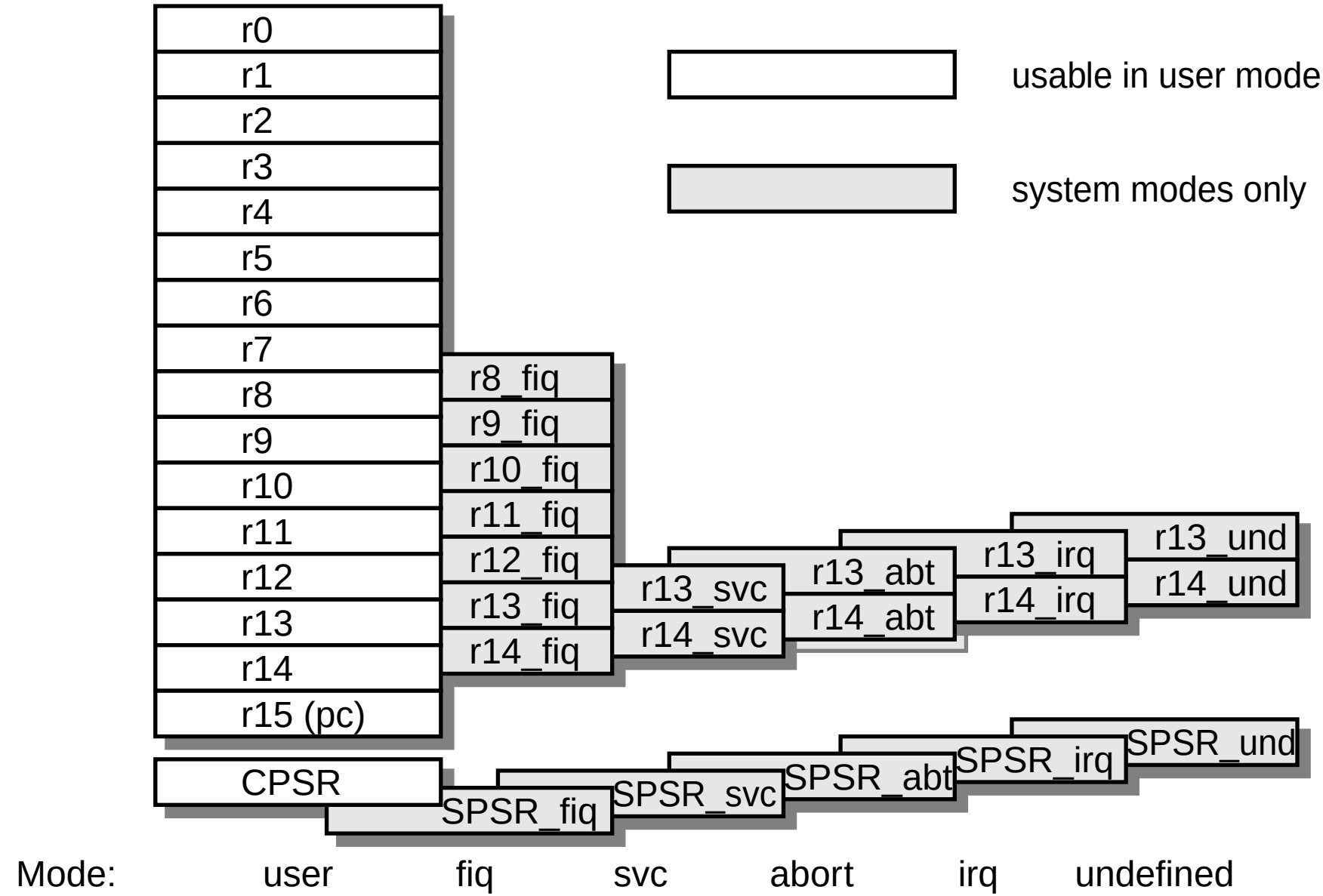
MANCHEstER 1824

The University of Manchester

Privileged modes and exceptions

- Each privileged mode (apart from System mode) has:
  - some private registers
    - its own r14 for a return address
    - its own r13, normally for a private stack pointer
    - FIQ mode has additional private registers to speed its operation
  - its own Saved Program Status Register (**SPSR**)
    - to preserve the CPSR so it can be restored upon return

Privileged modes and exceptions



Privileged modes and exceptions

- The CPSR and SPSR format:
  - bit 7 *disables* IRQ when set
  - bit 6 *disables* FIQ when set
  - bit 5 controls the instruction set
    - ARM (T=0) or Thumb (T=1)
  - bits 4 to 0 define the operating mode

Some of the “unused” bits have functions in later ARM versions.

Privileged modes and exceptions

- Register use:

CPSR [4:0]	Mode	Use	Registers
10000	User	Normal user code	user
10001	FIQ	Processing fast interrupts	_fiq
10010	IRQ	Processing standard interrupts	_irq
10011	SVC	Processing software interrupts (SWIs)	_svc
10111	Abort	Processing memory faults	_abt
11011	Undef	Handling undefined instruction traps	_und
11111	System	Running privileged operating system tasks	user

  - there is one more mode in a *few* recent ARMs (introduced later)

Privileged modes and exceptions

- Exceptions arise:
  - as a direct effect of fetching or decoding an instruction:
    - software interrupts
    - undefined instructions
    - prefetch aborts
  - as a side-effect of an instruction:
    - aborts on data transfers
  - unrelated to the instruction flow:
    - Reset, IRQ, FIQ

MANCHEstER 1824

The University of Manchester

Privileged modes and exceptions

- Exception entry sequence:
  - change to the appropriate operating mode
  - save the return address in r14_exc
  - save the old CPSR in SPSR_exc
  - disable IRQ
  - on FIQ entry, disable FIQ
  - force the PC to the appropriate exception ‘vector’ address
    - these are not really vectors!

Privileged modes and exceptions

- Exception vector addresses:

Exception	Mode	Vector address
Reset	SVC	0x00000000
Undefined instruction	UND	0x00000004
Software interrupt (SWI)	SVC	0x00000008
Prefetch abort (instruction fetch memory fault)	Abort	0x0000000C
Data abort (data access memory fault)	Abort	0x00000010
–	–	0x00000014
IRQ (normal interrupt)	IRQ	0x00000018
FIQ (fast interrupt)	FIQ	0x0000001C

Privileged modes and exceptions

- Exception handling
  - the ‘vector’ address normally contains a branch to the exception handling code

```
        B       exception_handler
or      LDR     PC, =exception_handler
```

    - the FIQ handler can start at 0x0000001C (saves branching)
  - r13_exc usually points to a private stack
    - can save work registers for use by the handler
    - FIQ usually has enough private registers
  - process exception
  - restore work registers and return

MANCHEstER 1824

The University of Manchester

Privileged modes and exceptions

- Privileged operations

```
MSR    CPSR_c, <source>; Change processor mode

STMFD  sp!, {r0-r15}^  ; Save user mode regs.
LDMFD  sp!, {r0-r14}^  ; Load user mode regs.
```

  - can also switch into system mode

```
LDMFD  sp!, {r15}^     ; Pop PC and restore SPSR
```

- Also in ARM v6

```
SRSFD  #svc!           ; Push LR and SPSR @SP_svc
RFEFD  SP!             ; Pop PC and CPSR
```

Privileged modes and exceptions

- Return from exception
  - from a SWI or undefined instruction:

```
MOVS pc, r14
```

    - data ops with S and pc are a special form
    - they restore the CPSR from SPSR_exc as well
  - from an IRQ, FIQ or prefetch abort:

```
SUBS pc, r14, #4
```

  - from a data abort to retry the data transfer:

```
SUBS pc, r14, #8
```

Example interrupt handler

```
00000014      ...
00000018      B     irq_handler          ; “Vector”
0000001C      ...

irq_handler   SUB   lr, lr, #4           ; ‘correct’ return addr.
              STMFD SP!, {r0-r2, lr}     ; Save working regs.
              ...                        ; Determine IRQ source
              ...                        ; Branch to ISR


              ...
              LDMFD SP!, {r0-r2, pc}^ ; Restore and return
```

- using a Branch at the ‘vector’ position
  - limits range of jumps
  - ‘tedious’ to modify

MANCHEstER 1824

The University of Manchester

Example data abort handler

```
0000000C        ...
00000010        ldr   pc, dabt_vector   ; “Vector”
00000014        ...

                ...
dabt_vector     DCD   dabt_handler      ; Address of routine
                ...
dabt_handler    SUB   lr, lr, #8        ; ‘correct’ return addr.
                STMFD SP!, {r0-r2, lr}  ; Save working regs.
                LDR   r14, [lr]         ; Get failed instruction
                ...
                ...                     ; Find failure address
                ...
                ...                     ; (e.g.) load page
                ...
                LDMFD SP!, {r0-r2, pc}^ ; Restore and return
```

MANCHEstER 1824

The University of Manchester

The ARM instruction set

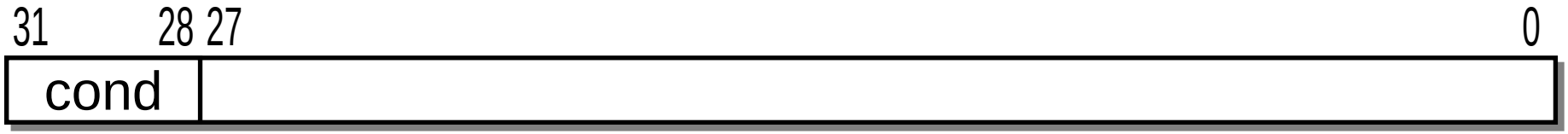
- Outline:
  - privileged modes and exceptions
  - ➜ **instruction set details**
  - system code example

  - ☞ hands-on: system software - SWI handler

MANCHEstER 1824

The University of Manchester

The ARM condition code field



- (almost) every ARM instruction may have a condition added
  - exceptions (later versions) use former ‘NV’ code for ‘always’
  - the instruction will only be executed if the condition is passed
  - the conditions test the values of the N, Z, C and V flags in the CPSR
- if no condition is specified ‘AL’ (always) is assumed

ARM condition codes

Opcode [31:28]	Mnemonic extension	Interpretation	Status flag state for execution
0000	EQ	Equal/equals zero	Z set
0001	NE	Not equal	Z clear
0010	CS/HS	Carry set/unsigned Higher or same	C set
0011	CC/LO	Carry clear/unsigned lower	C clear
0100	MI	Minus/negative	N set
0101	PL	Plus/positive or zero	N clear
0110	VS	Overflow	V set
0111	VC	No Overflow	V clear
1000	HI	Unsigned higher	C set and Z clear
1001	LS	Unsigned lower or same	C clear or Z set
1010	GE	Signed greater or equal	N equals V
1011	LT	Signed less than	N is not equal to V
1100	GT	Signed greater than	Z clear and N equals V
1001	LE	Signed less or equal	Z set or N is not equal to V
1110	AL	Always	any
1111	NV†	Never	none

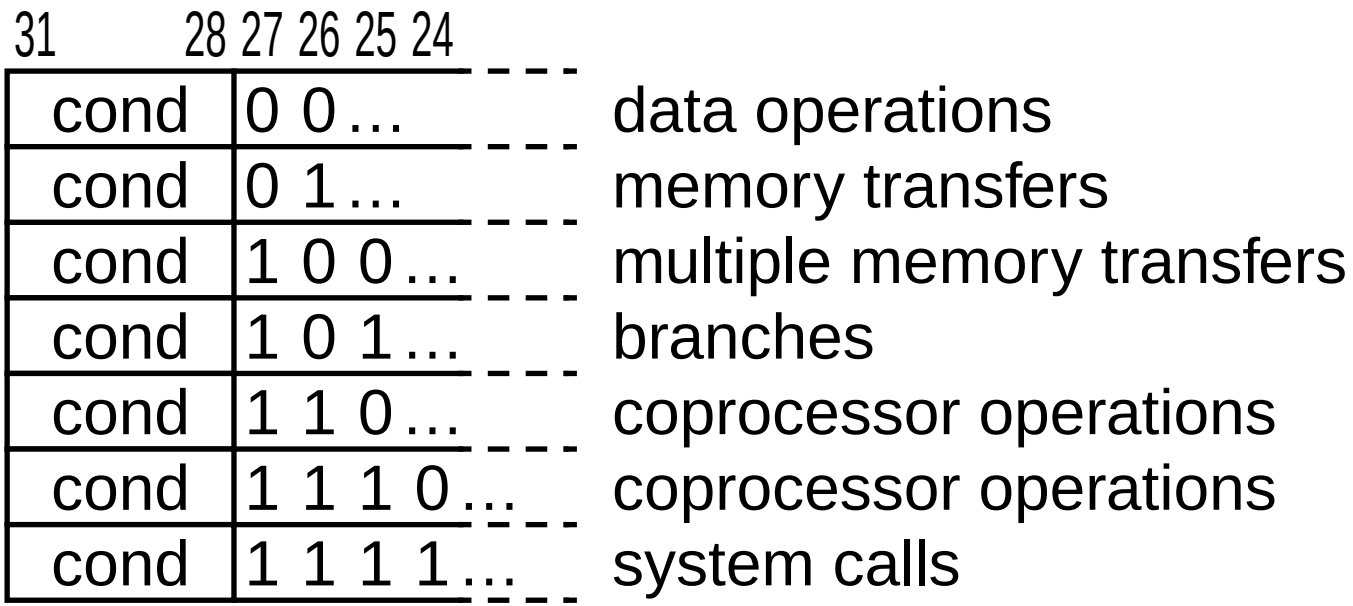
†NV (1111) is used to specify other, unconditional instructions in later ARM versions.

MANCHEstER 1824

The University of Manchester

ARM instruction format

- All ARM instructions are 32 bits long
- Originally the decoding was quite simple



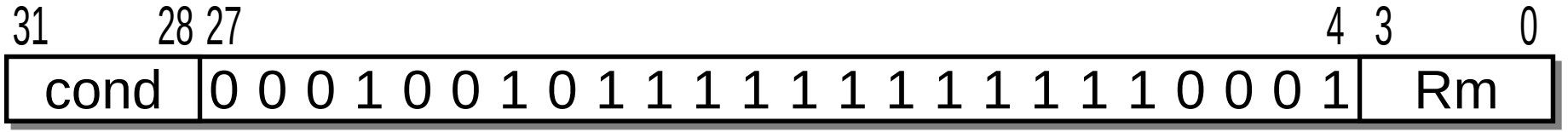
  - the model is no longer quite this simple!
  - ‘holes’ in the instruction space have since been filled

Branch and Branch with Link

- the L bit selects Branch with Link
  - the address of the instruction after the branch is placed into r14
- the offset is scaled to word
  - giving a range of ±32 Mbytes
- Assembler format:

```
B{L}{<cond>} <target address>
```

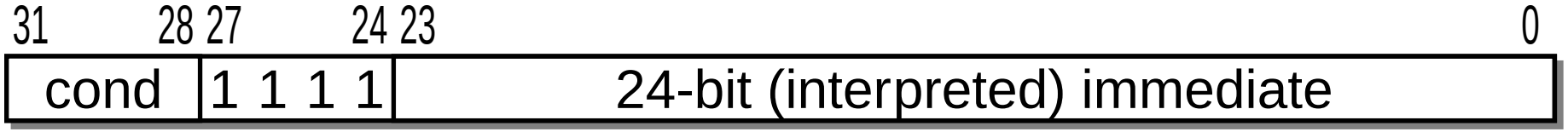
Branch and eXchange



- recent ARM chips (v5T) also support BLX
- used to switch execution to the Thumb instruction set
  - if Rm[0] = 1
- causes a branch to the address in Rm
  Assembler format:

```
BX{<cond>} Rm
```

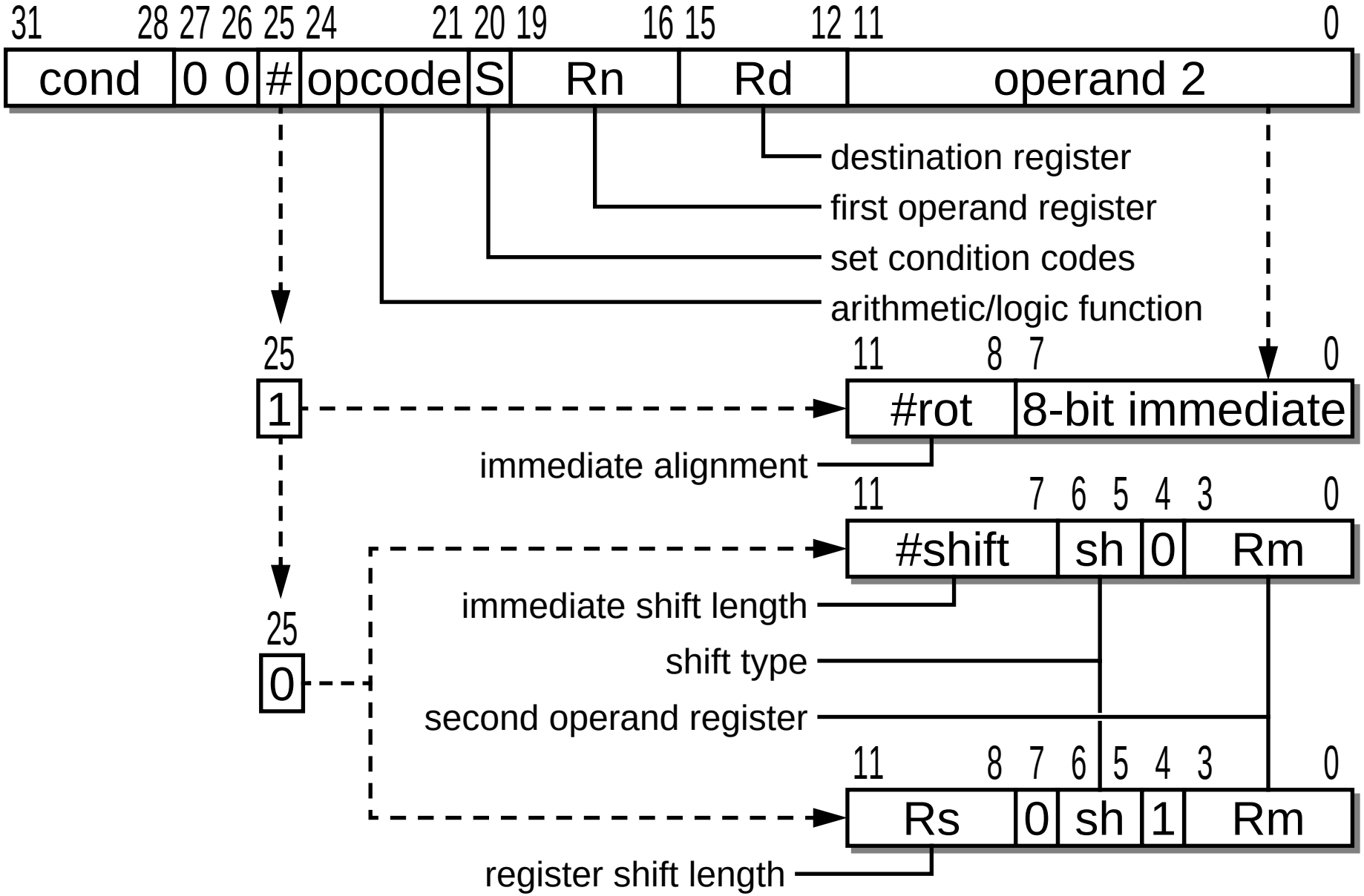
SoftWare Interrupt



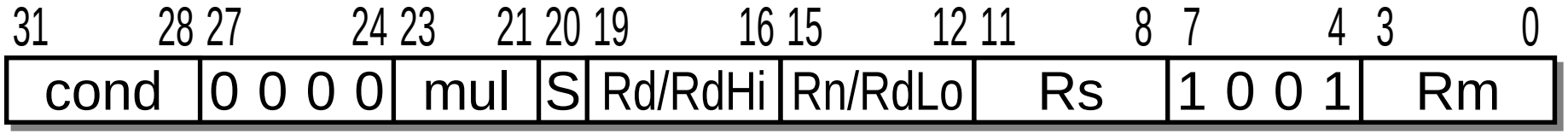
- this instruction is the normal way to access operating system facilities; it:
  - puts the processor into supervisor mode
  - saves the CPSR in SPSR_svc
  - saves the return address in r14_svc
  - sets the PC to 0x00000008
- Assembler format:

```
SWI{<cond>} <24-bit immediate>
```

Data processing instructions



Multiply instructions

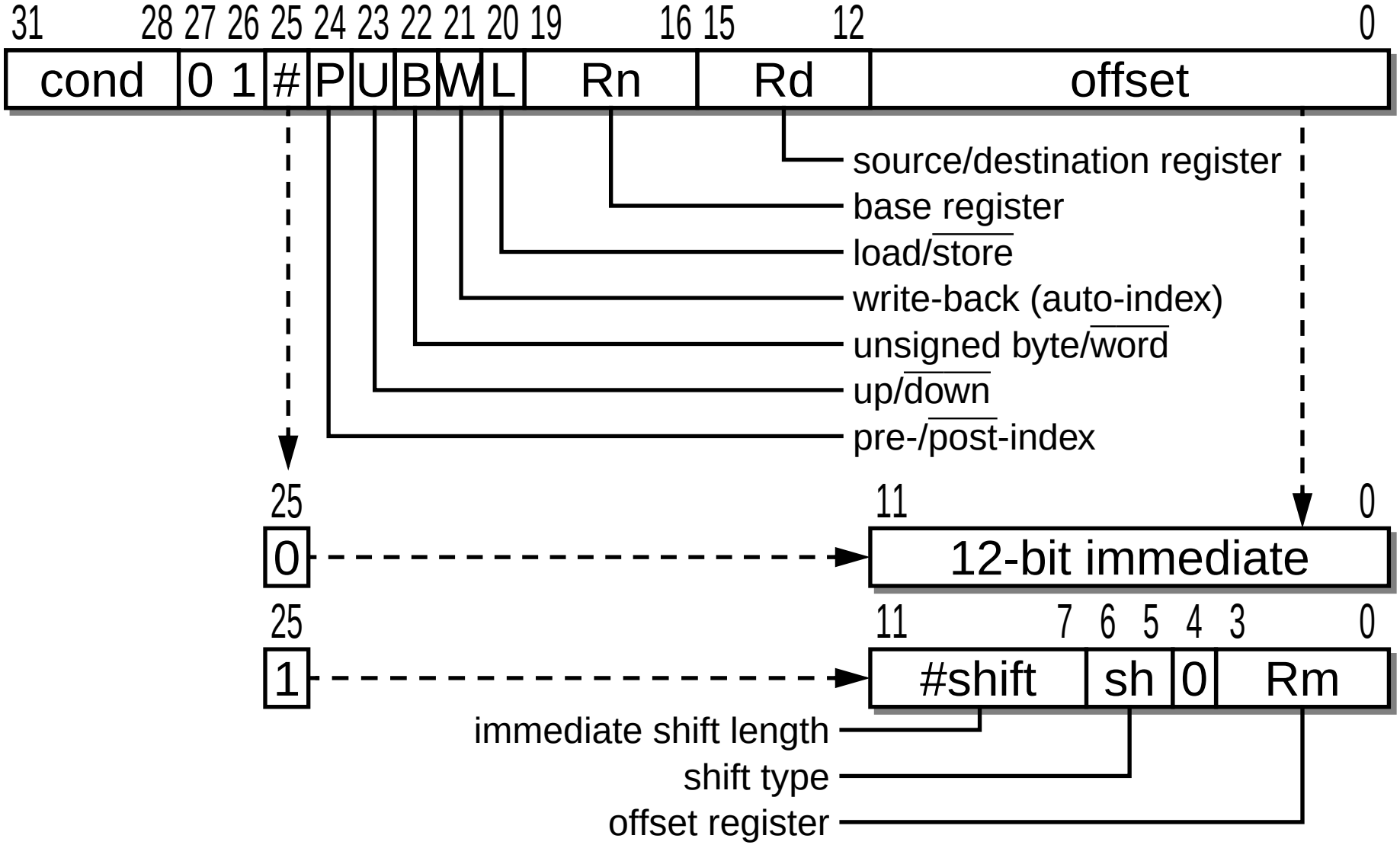



```
MUL{<cond>}{S}    Rd, Rm, Rs
MLA{<cond>}{S}    Rd, Rm, Rs, Rn
<mul>{<cond>}{S} RdHi, RdLo, Rm, Rs
```

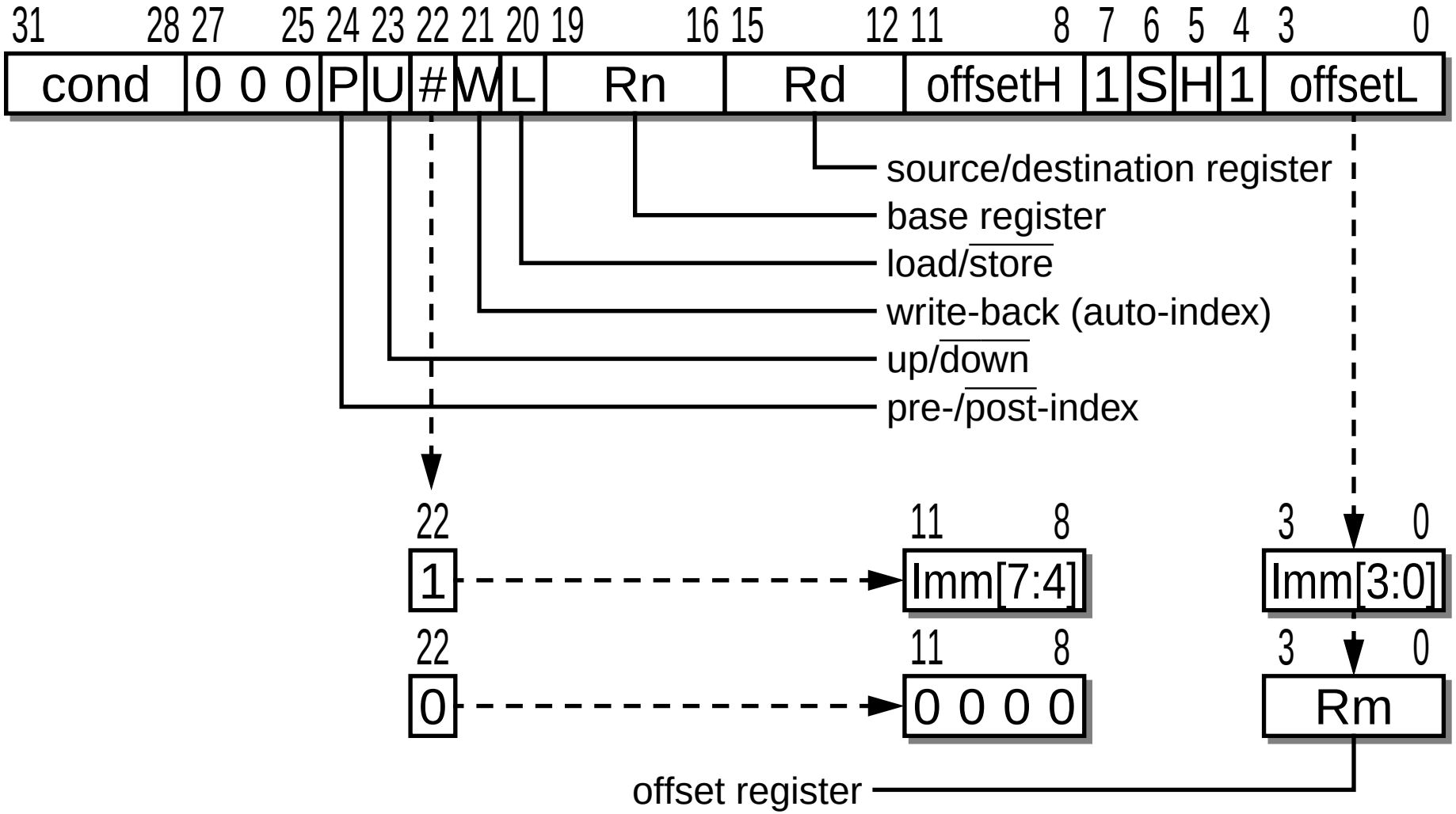
Opcode [23:21]	Mnemonic	Meaning	Effect
000	MUL	Multiply (32-bit result)	Rd := (Rm * Rs) [31:0]
001	MLA	Multiply-accumulate (32-bit result)	Rd := (Rm * Rs + Rn) [31:0]
010	UMAAL*	Unsigned multiply-accumulate-accumulate long	RdHi:RdLo := Rm * Rs + RdHi + RdLo
011	–	<unused>	–
100	UMULL	Unsigned multiply long	RdHi:RdLo := Rm * Rs
101	UMLAL	Unsigned multiply-accumulate long	RdHi:RdLo += Rm * Rs
110	SMULL	Signed multiply long	RdHi:RdLo := Rm * Rs
111	SMLAL	Signed multiply-accumulate long	RdHi:RdLo += Rm * Rs

*UMAAL was introduced in ARM v6

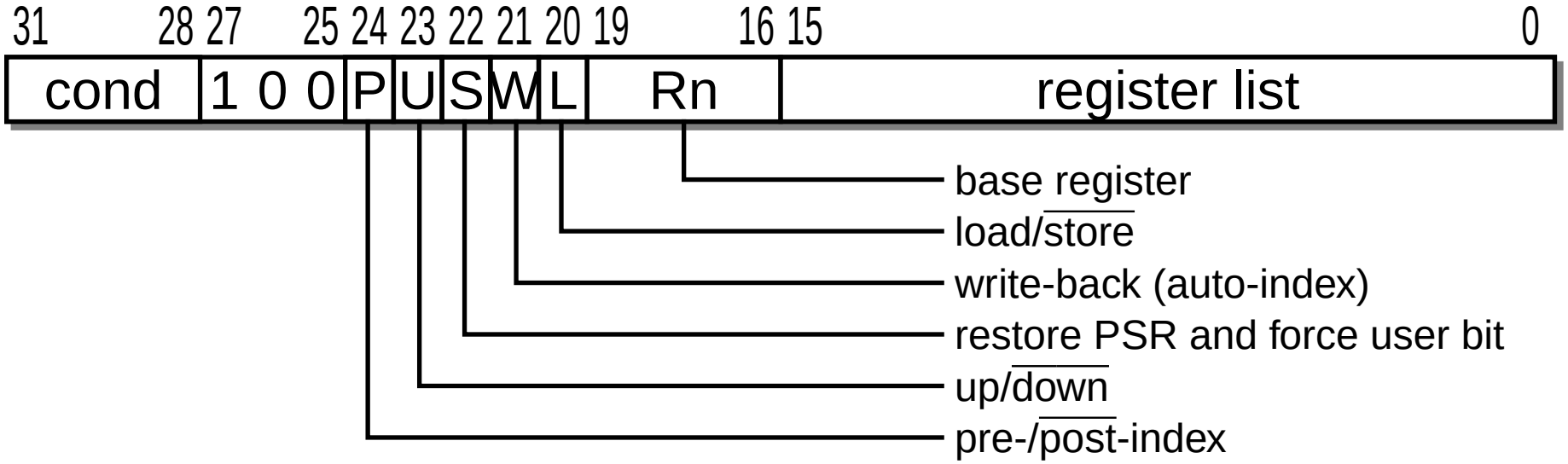
Single word and unsigned byte data transfer instructions



Half-word and signed byte data transfer instructions



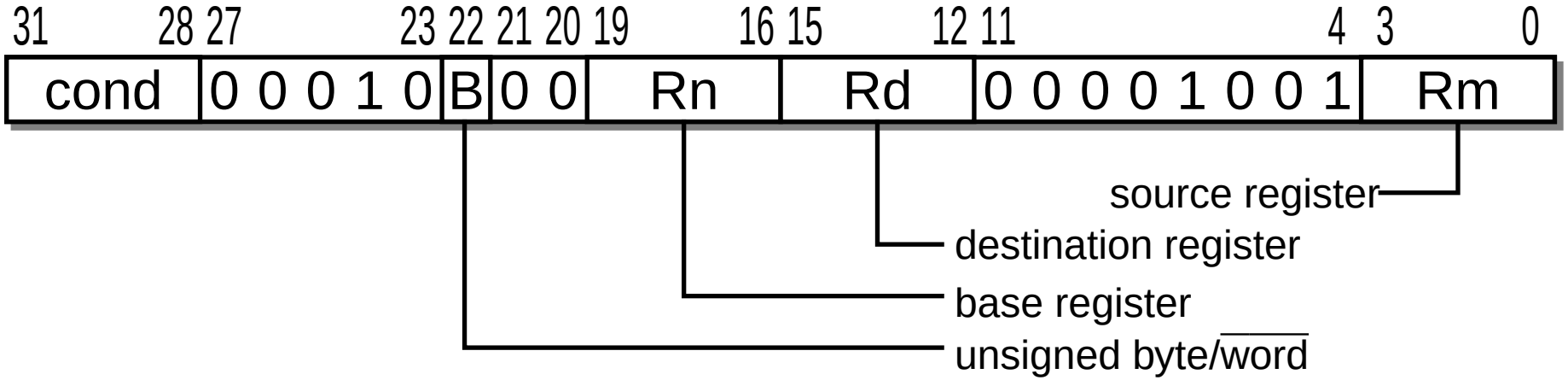
Multiple register data transfers



- Assembler format:

```
LDM|STM{<cond>}<add> Rn{!},<regs>
<add> = IA etc, <regs> = {rn,..rm}
```

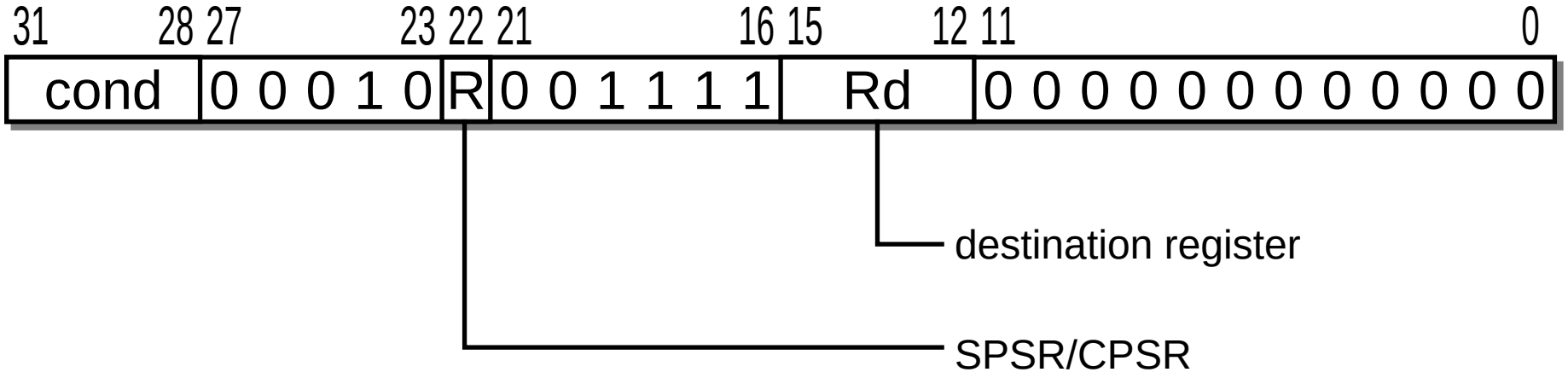
Swap memory and register



- Assembler format:

```
SWP{<cond>}{B} Rd,Rm,[Rn]
```

Status register to general register



- Assembler format:

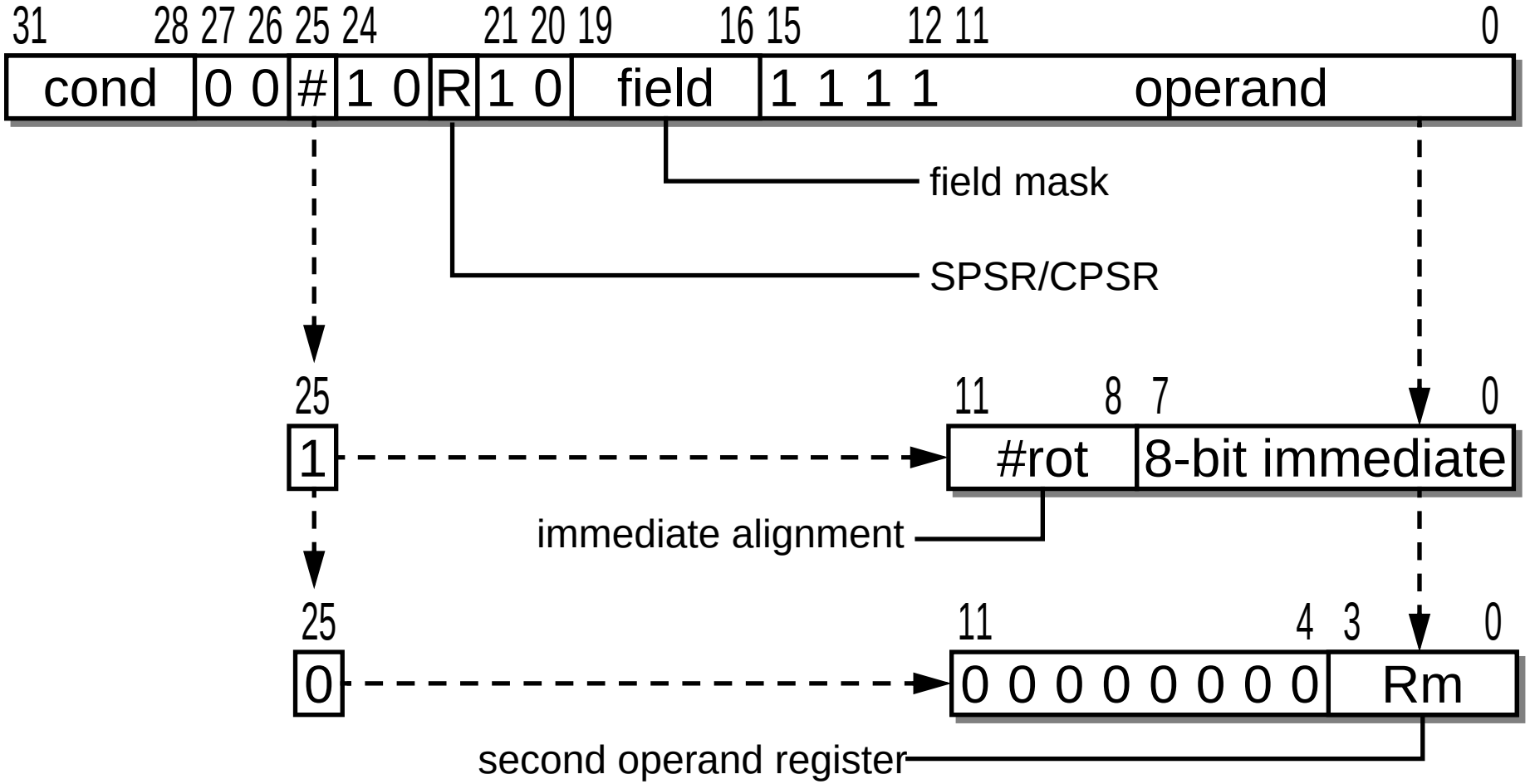
```
MRS{<cond>} Rd, CPSR|SPSR
```

- and the reverse (see next slide):

```
MSR{<cond>} CPSR|SPSR, #32|Rm
```

- (with a few details about fields omitted)

Transfers to status register



MANCHEstER 1824

The University of Manchester

The ARM instruction set

- Outline:
  - privileged modes and exceptions
  - instruction set details
  - ➜ **system code example**

  - ☞ hands-on: system software - SWI handler

MANCHEstER 1824

The University of Manchester

System code example

- Process swap code (for 2 processes)
  - save full processor state (ARM or Thumb)
  - restore alternate state
  - switch process on interrupt
    - e.g. from Timer for pre-emptive scheduling
  - uses ‘force user mode’ form of LDM/STM
    - has restrictions on base register write-back, inclusion of r15, and so on
    - some ARM cores require 1 cycle delay before a banked register may be used after ‘force user’

MANCHEstER 1824

The University of Manchester

Process state data structure

```
r14Temp   DCD   0x0              ; r14 temp. save area
procNo    DCD   0x0              ; current process ID
procTab   DCD   proc0save        ; -> save areas
          DCD   proc1save
proc0save DCD   0x00000000       ; pc
          DCD   0x00000000       ; CPSR
          DCD   0x00000000       ; r0
          DCD   0x00000000       ; r1
          ...
          DCD   0x00000000       ; r14
proc1save DCD   proc1            ; pc
          DCD   0x10             ; CPSR
          DCD   0x00000000       ; r0
          DCD   0x00000000       ; r1
          ...
          DCD   0x00000000       ; r14
```

MANCHEstER 1824

The University of Manchester

Process save code

```
; save user process
SUB    r14, r14, #4    ; adjust IRQ return address
STR    r14, r14Temp    ; temporary save of r14
LDR    r13, procNo     ; which process is running?
ADR    r14, procTab
LDR    r13, [r14,r13,LSL #2]
LDR    r14, r14Temp    ; restore r14
STMIA  r13!,{r14}      ; user prog. return address
MRS    r14, SPSR
STMIA  r13!,{r14}      ; save user CPSR
STMIA  r13,{r0-r14}^   ; force user mode [no pc/WB]
```

Process resume code

```
; restore other user process
LDR    r13, procNo     ; which process is running?
RSB    r13, r13, #1    ; other process number
STR    r13, procNo     ; change stored process ID
ADR    r14, procTab
LDR    r13, [r14,r13,LSL#2]
LDMIB  r13!,{r14}      ; get user CPSR
MSR    SPSR, r14
LDMIB  r13,{r0-r14}^   ; force user mode [no pc/WB]
MOV    r0, r0          ; NOOP after force user
LDMDB  r13,{pc}^       ; restore CPSR and pc
```

MANCHEstER 1824

The University of Manchester

Hands-on: system software – SWI handler

- Look at ARM system software programs
  - Write a SWI handler
  - Check that it works as expected

☞ Follow the ‘Hands-on’ instructions