## The ARM instruction set

- Outline:
  - privileged modes and exceptions
  - instruction set details



## The ARM instruction set

- Outline:
  - ➔ **privileged modes and exceptions**
  - instruction set details



## 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



## 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

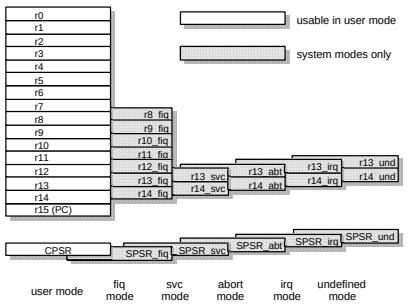


## Privileged modes and exceptions

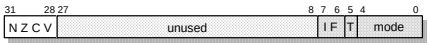
- Each privileged mode (apart from System mode) has:
  - some private registers
    - its own r14 for a return address
    - its own r13, normally for a private stack pointer
    - FIQ mode has additional private registers to speed its operation
  - its own Saved Program Status Register (**SPSR**)
 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



## 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 |



## 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



## 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 |
| 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
    - the FIQ handler can start at 0x1C
  - 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

## 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
```

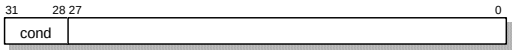


## The ARM instruction set

- Outline:
  - privileged modes and exceptions
  - ➔ **instruction set details**



## The ARM condition code field




- every ARM instruction may have a condition added
  - 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

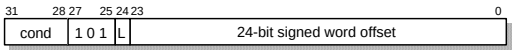


## ARM condition codes

| 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 than 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 |
| 1101 | LE | Signed less than or equal | Z set or N is not equal to V |
| 1110 | AL | Always | any |
| 1111 | NV | Never (do not use!) | none |



## Branch and Branch with Link



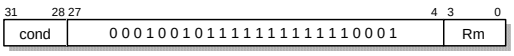

- the L bit selects Branch with Link
  - the address of the instruction after the branch is placed into r14
- the offset is scaled to word
  - giving a range of +/- 32 Mbytes

Assembler format:

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



## Branch and eXchange

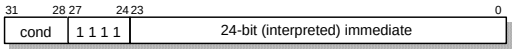



- recent ARM chips (v5T) also support BLX
- used to switch execution to the Thumb instruction set
  - if Rm[0] = 1
- causes a branch to the address in Rm

Assembler format:

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

# SoftWare Interrupt



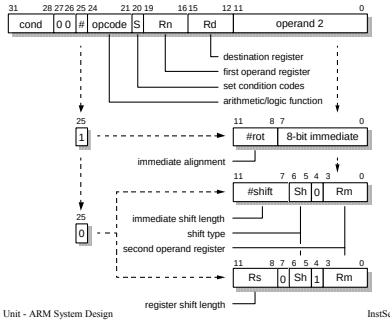

- this instruction is the normal way to access operating system facilities; it:
  - puts the processor into supervisor mode
  - saves the CPSR in SPSR_svc
  - saves the return address in r14_svc
  - sets the PC to 0x8

Assembler format:

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

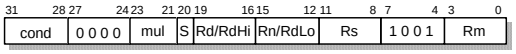


# Data processing instructions






# Multiply instructions

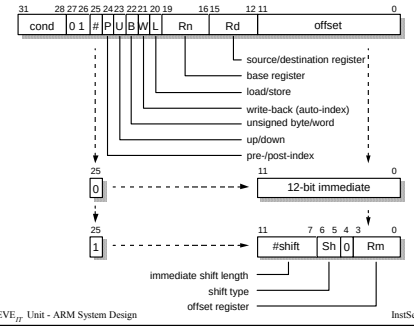



```
MUL{<cond>}{S}   Rd, Rm, Rs
MLA{<cond>}{S}   Rd, Rm, Rs, Rn
<mul>{<cond>}{S} RdHi, RdLo, Rm, Rs
```

| 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] |
| 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 |

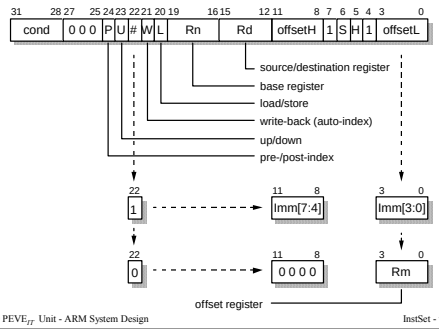


# Single word and unsigned byte data transfer instructions



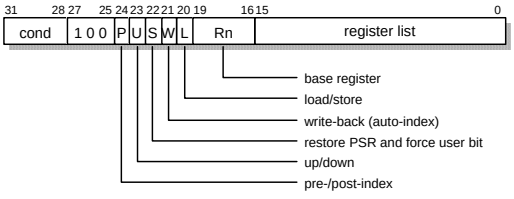



# Half-word and signed byte data transfer instructions






# Multiple register data transfers




Assembler format:

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