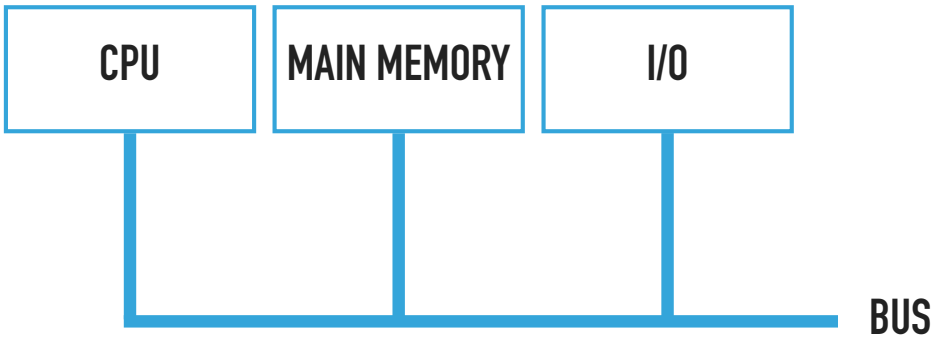
```
15 .global _start
16 _start:
17     mov sp, #0x8000000
18     push {ip, lr}
19
20     bl led_pin_enable
21
22 1:  bl led_on
23     mov r0, #0x00400000 /* duratio
24     bl delay_us
25     bl led_off
26     mov r0, #0x00100000 /* duratio
27     bl delay_us
28     b 1b
29
30 /* led_pin_enable
31  * args: none
32  */
33 led_pin_enable:
34     ldr r0,=GPFSEL2
35     ldr r0,[r0]
36     bic r1, r0, #0x38000000
37     orr r1, r1, #0x08000000
38     ldr r0,=GPFSEL2
39     str r1, [r0]
40     bx lr
41
42 /* led_on
43  *
44  * args: none
45  */
46 led_on:
47     ldr r0,=GPSET0
48     mov r1, #0x20000000  /* GPIO p
```

EMBEDDED SYSTEMS
2020/2021

ISA LEVEL

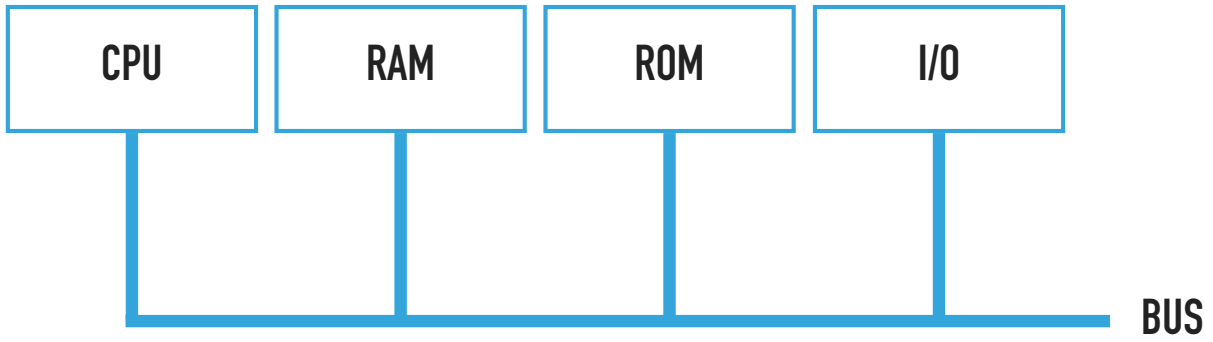
ISA LEVEL

- In the Von Neumann Machine, at the ISA Level the CPU executes Machine Language (ML) programs stored into the Main Memory
  - ML programs are sequences of instructions to be natively executed one after another by the CPU
  - Each ML instruction is stored as specific configuration of bits
  - ML instructions are stored compactly according to the execution order
  - ML instructions can drive I/O devices to obtain (INPUT) or provide (OUTPUT) data in human-readable form



ISA LEVEL

- In actual machines more than one memory can be found
  - The largest is usually the Random Access Memory, which is “volatile"
  - Non-volatile Read Only Memories (ROM) are included to provide the CPU with programs at power-on
  - Other kinds of memory can also be available



ISA LEVEL

- The Main Memory is organized as an array of cells containing the same number of bits
- In most cases each memory cell contains a byte (eight bits)
- Each cell is accessed by a number called **address**
  - On the right: a 64KB (65536 bytes) memory requires a 16-bit number (2^16=65536) to address a cell

Address	Value
0000	00
0001	FF
0002	27
0003	1B
0004	02
…	…
0400	03
0401	09
0402	01
0403	0F
…	…
FFFE	0C
FFFF	80

MULTIPLE-BYTE VALUES

- Contiguous cells can be grouped to store multiple-byte values
  - Content of cells 0002 and 0003 could be interpreted as a 16-bit value
  - Content of cells 0400-0403 could be interpreted as a 32-bit value
- But how?
  - Considering the addresses from the lowest to the highest
    - the **little-endian** representation encodes the multiple-byte value from the least significant byte (LSB), the **little end**, to the most significant byte (MSB) (eg. 1B27 and 0F010903)
    - the **big-endian** representation uses the opposite ordering so that multiple-byte values are read from the MSB, the **big end,** to the LSB (e.g 271B and 0309010F)
- Memory content is usually **dumped** as addresses grow
  - the **big-endian** representation is easier to decode for humans
  - However, most architectures adopt the **little-endian** representation

Address	Value
0000	00
0001	FF
0002	27
0003	1B
0004	02
…	…
0400	03
0401	09
0402	01
0403	0F
…	…
FFFE	0C
FFFF	80

ADDRESS DECODING

- Consider a system with an n-bit address bus
- The full range of 2^n addresses must be shared among all the components connected to the bus
- Properly sized components must be chosen
- A decoding circuitry must be present using, for instance a Chip-Select (CS) signal connected to the same-named pin on each component chip
- More sophisticated schemes (e.g. programmable decoders or memory-management units) support overlapping address ranges for components or virtual address spaces even larger than what the physical bus would allow

Address	Value	Chip
0000	00	16 KB RAM
…	…	
3FFF	34	
…	UNUSED	
8000	40	8KB ROM
…	…	
9FFF	62	
…	UNUSED	
FFFC	0F	4B I/O
FFFD	0C	
FFFE	0C	
FFFF	80	

THE ARM INSTRUCTION SET

- ARM instruction set formats
  - Some formats define more than one operation

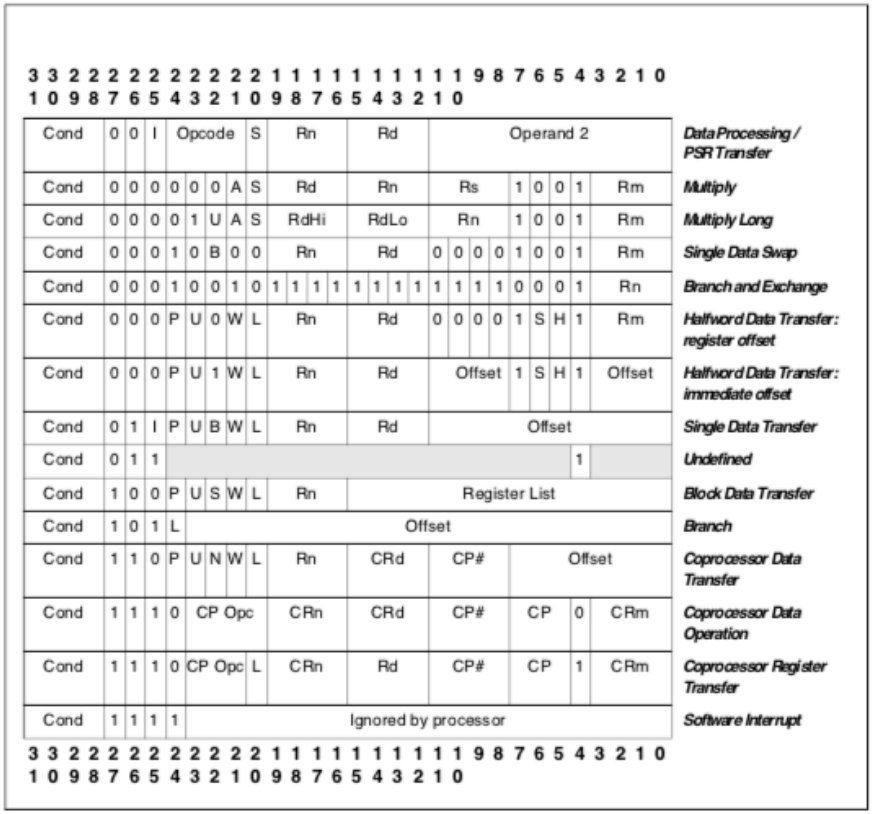


Figure 4-1: ARM instruction set formats

THE ARM INSTRUCTION SET

- Data processing instruction format
  - Many data processing operations can be interpreted as being executed by a single flexible instruction
  - The instruction performs a specified arithmetic or logical operation on one or two operands
  - The first operand, if present, is always a register (Rn)
  - In the MOV and MVN variants the first operand is ignored

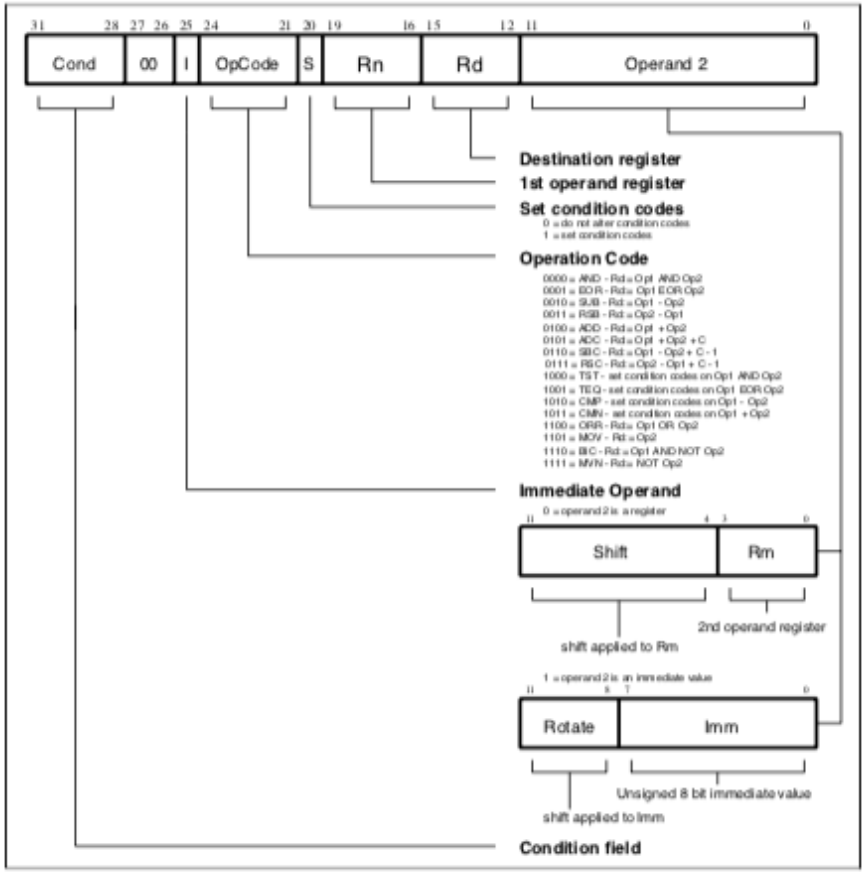


Figure 4-4: Data processing instructions

ARM DDI 0084D - ARM Instruction Set - p. 4-10 - (arm-instructionset.pdf)