2.0 Instruction Set Architecture (ISA)

Instruction Set Architecture of a processor specifies:

- Instruction set (syntax and semantic)
- Data operand location
- Registers
  - How many
  - How to reference
  - How to use

➢ ISA viewed as:

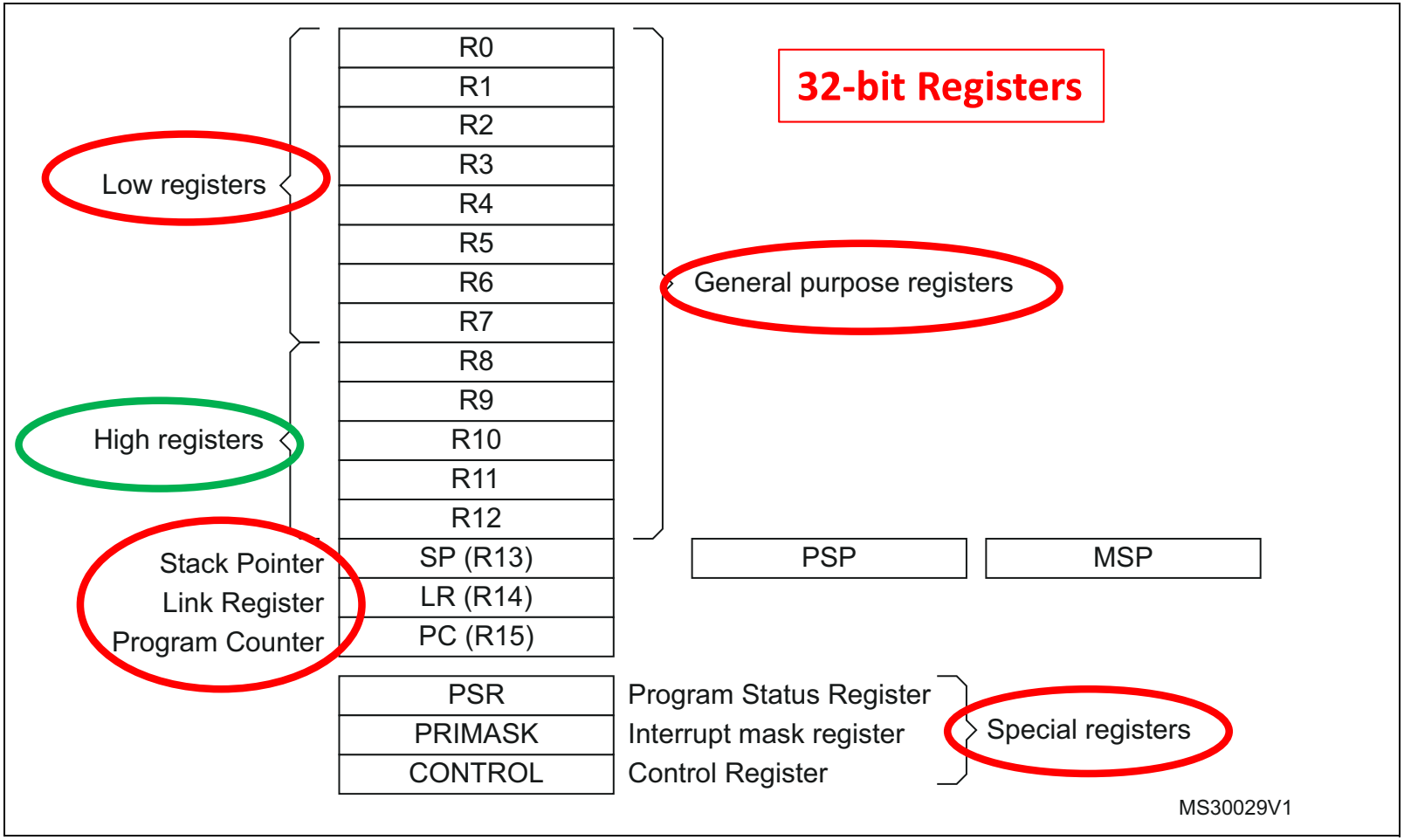
  ➢ User-Machine interface

  ➢ Hiding hardware details

ARM Registers (PM12)

(Page-12 Programming Manual)

Figure 2. Processor core registers



ARM Register Usage (PM13)

Stack pointer (SP) register R13

Operating System

In Thread mode, bit[1] of the CONTROL register indicates the stack pointer to use:

- 0: Main Stack Pointer (MSP)(reset value). On reset, the processor loads the MSP with the value from address 0x00000000.
- 1: Process Stack Pointer (PSP).

User Program

Link register (LR) register R14

Stores return information for subroutines, function calls, and exceptions. On reset, the processor loads the LR value 0xFFFFFFFF.

Program counter (PC) register R15

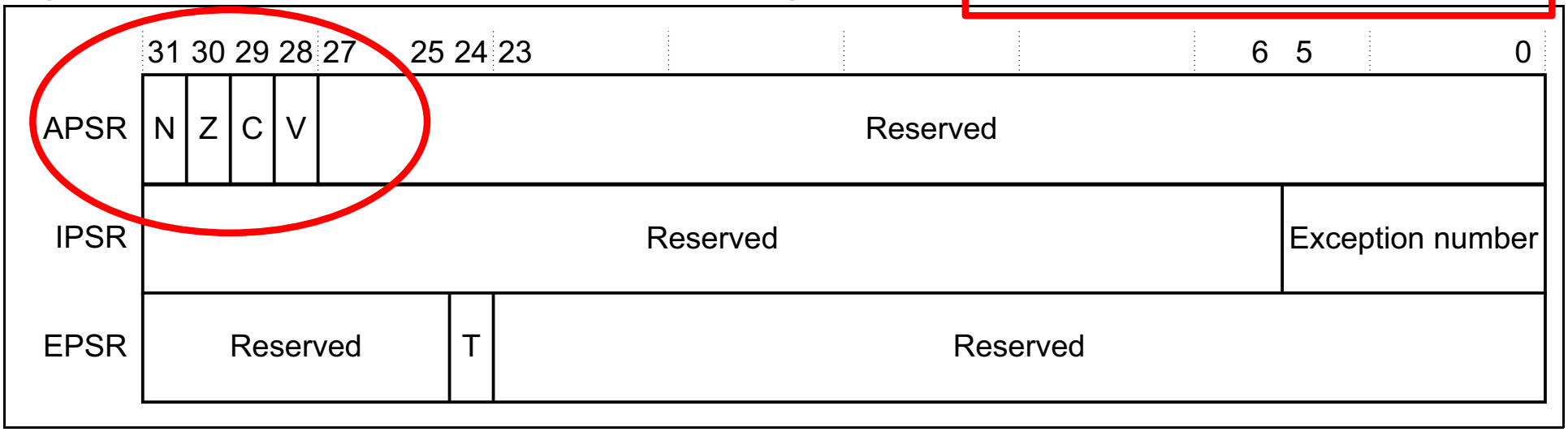
Contains the current program address. On reset, the processor loads the PC with the value of the reset vector, which is at address 0x00000004. Bit[0] of the value is loaded into the EPSR T-bit at reset and must be 1.

ARM Program Status Register (PM13)

Next instructions: to execute or not? By checking condition bits

Figure 3. APSR, IPSR and EPSR bit assignments

Conditional Execution!



Result of operation is

N = Negative

6 - 9 = -3

6 - 6 = 0

Z = Zero

6 + 6 = 12

C = Carry

6 + 3 = 9_{10} in 3 binary bits?

V = Overflow

2.1 Instruction Encoding-1

- Example, 16-bit instruction:
  - Move R1,R2

operation	source	destination
(Move opcode) 011001	001	010

    - Given: 6-bit opcode, two 3-bit register numbers (R0…R7)
    - Total: 12 bits
    - Needs 2 bytes to accommodate 12 bits
      - 0000 0110 0100 1010

2.1 Instruction Encoding-2

- Example, Store R3,%00001111

operation	source	destination
(Store) 000111	011	00001111

  - 6-bit operation, one 3-bit register number; 8-bit address

    = 17 bits ; needs Two 16-bit words

Each machine instruction requires one or more consecutive words (for a 16-bit processor)

ARM 'ADD' Instruction (PM49)

{s} means s is optional

ADD{S} {Rd, } Rn, <Rm | #imm>

< x | y > means (x or y)

where:

- *S*: causes an ADD or SUB instruction to update flags
- *Rd*: specifies the result register. If omitted,this value is assumed to take the same value as *Rn*, for example ADDS R1,R2 is identical to ADDS R1,R1,R2.
- *Rn*: specifies the first source register
- *Rm*: specifies the second source register
- *imm*: specifies a constant immediate value.

The ADD instruction adds the value in *Rn* to the value in *Rm* or an immediate value specified by *imm* and places the result in the register specified by *Rd*.

The ADDS instruction performs the same operation as ADD and also updates the N, Z, C and V flags.

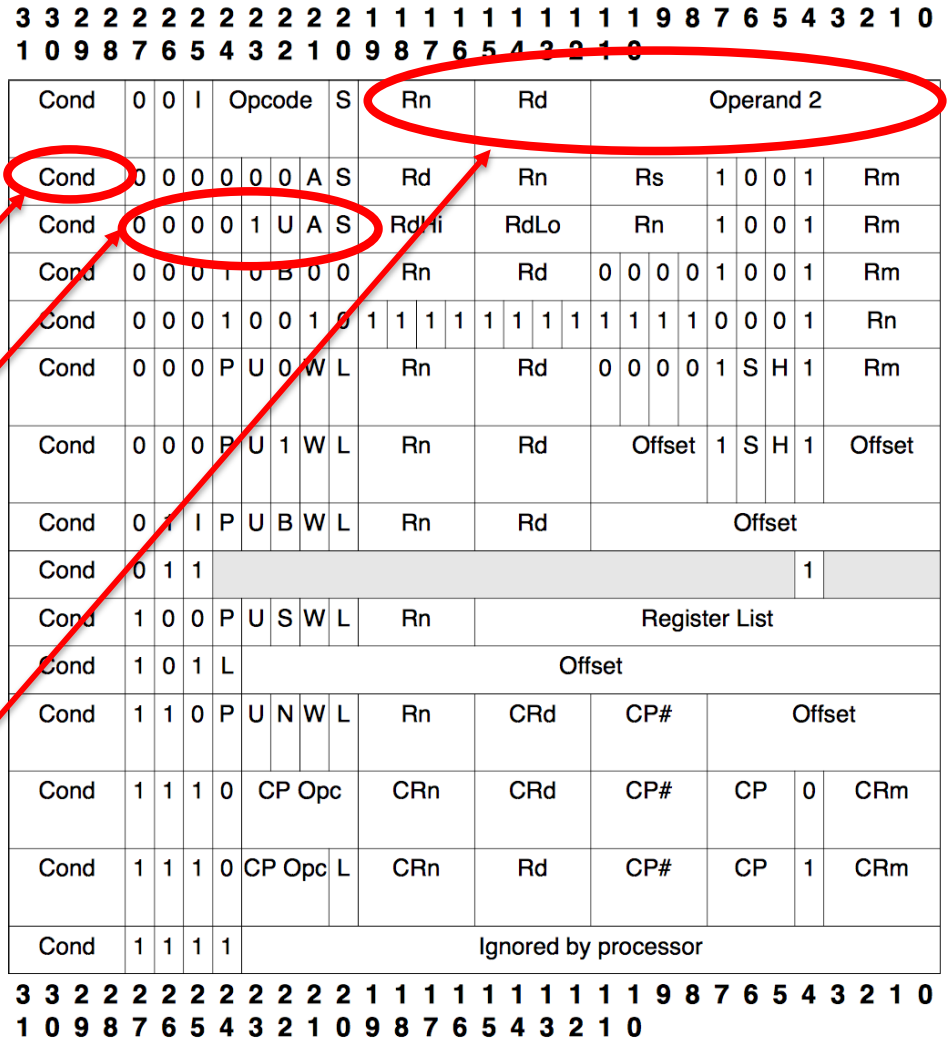
ARM Instruction Set Format

32-bit Instructions
$b_{31}…b_0$

Cond: N/Z/C/V
$b_{31}…b_{28}$

Operation
$b_{27}…b_{20}$

Operands
$b_{19}…b_0$



Total 57 instructions

Thumb ADD/SUB (TH5-7)

ARM7-TDMI-manual-pt3: Ch5 Sec7

??? Is a Subset of

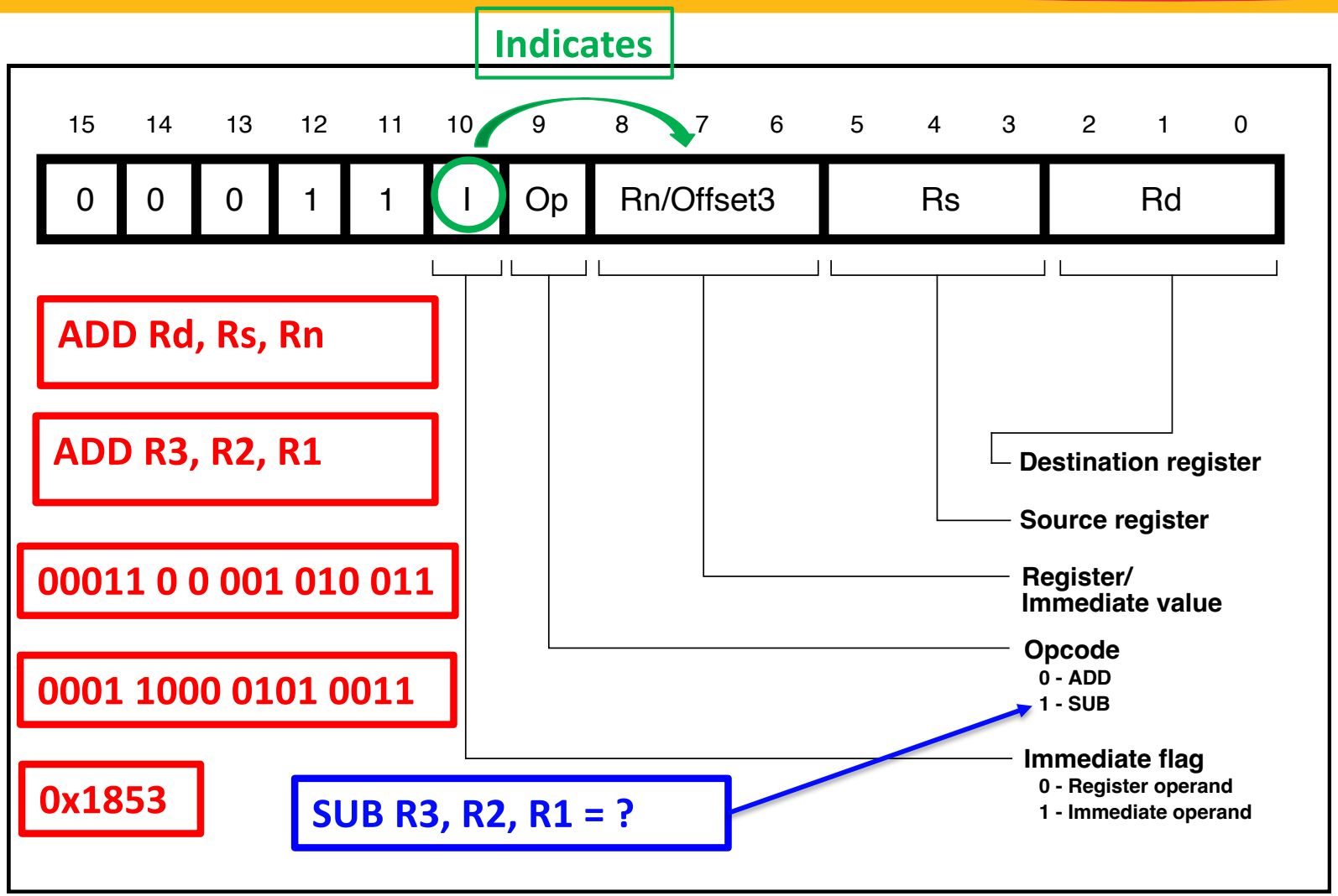
Op	I	THUMB assembler	ARM equivalent	Action
0	0	ADD Rd, Rs, Rn	ADDS Rd, Rs, Rn	Add contents of Rn to contents of Rs. Place result in Rd.
0	1	ADD Rd, Rs, #Offset3	ADDS Rd, Rs, #Offset3	Add 3-bit immediate value to contents of Rs. Place result in Rd.
1	0	SUB Rd, Rs, Rn	SUBS Rd, Rs, Rn	Subtract contents of Rn from contents of Rs. Place result in Rd.
1	1	SUB Rd, Rs, #Offset3	SUBS Rd, Rs, #Offset3	Subtract 3-bit immediate value from contents of Rs. Place result in Rd.

ADD R0, R3, R4 ; R0 := R3 + R4

Offset3 = {000_2 to 111_2} = {0_{10}..7_{10}}

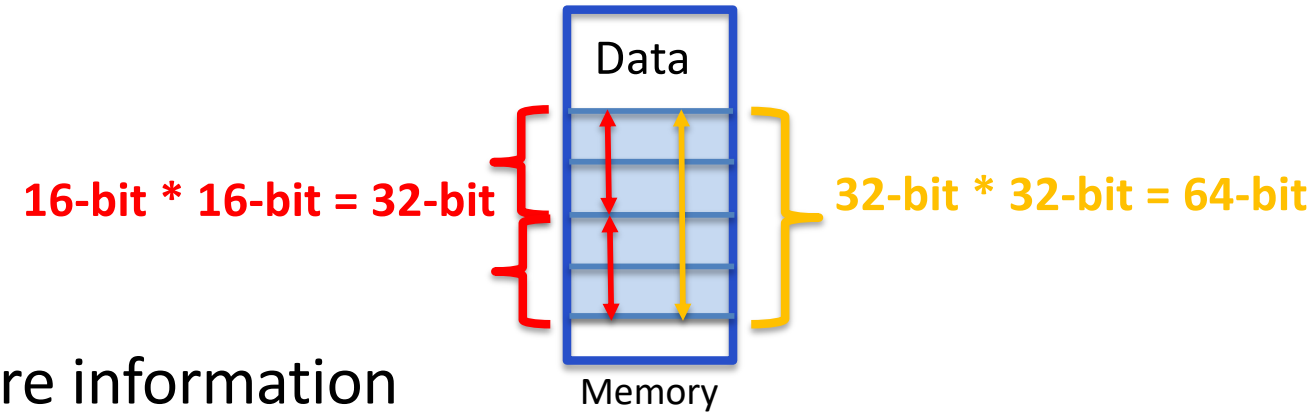
SUB R6, R2, #6 ; R6 := R2 - 6_{10}

Thumb ADD/SUB Encoding (TH5-7)



ARM & Thumb Architecture

- ARM: 32-bit RISC (Reduced Instruction Set Computer) architecture
  - 32-bit instructions and 32-bit data
- 16-bit vs 32-bit?
  - Data
    - More bits, more information
    - Better accuracy

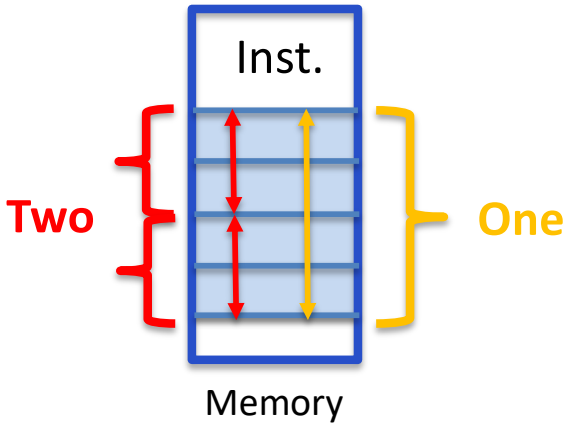


ARM & Thumb Architecture

- ARM: 32-bit RISC (Reduced Instruction Set Computer) architecture
  - 32-bit instructions and 32-bit data

- 16-bit vs 32-bit ?
  - instructions
    - Less bits, less space
    - Higher code density

      (amount of space a program takes in memory)



Thumb Instruction Set

- Subset of ARM: most commonly-used 32-bit ARM instructions
  - ARM's 57 instructions vs Thumb's 36
- Thumb instructions are 16 bits vs ARM's 32 bits
- Use low registers R0-R7
- During execution, 16-bit Thumb instructions are mapped to the corresponding full 32-bit ARM instructions, having:
  - 32-bit registers
  - 32-bit address space
  - 32-bit ALU
  - 32-bit memory transfer

Thumb Instruction Set Format

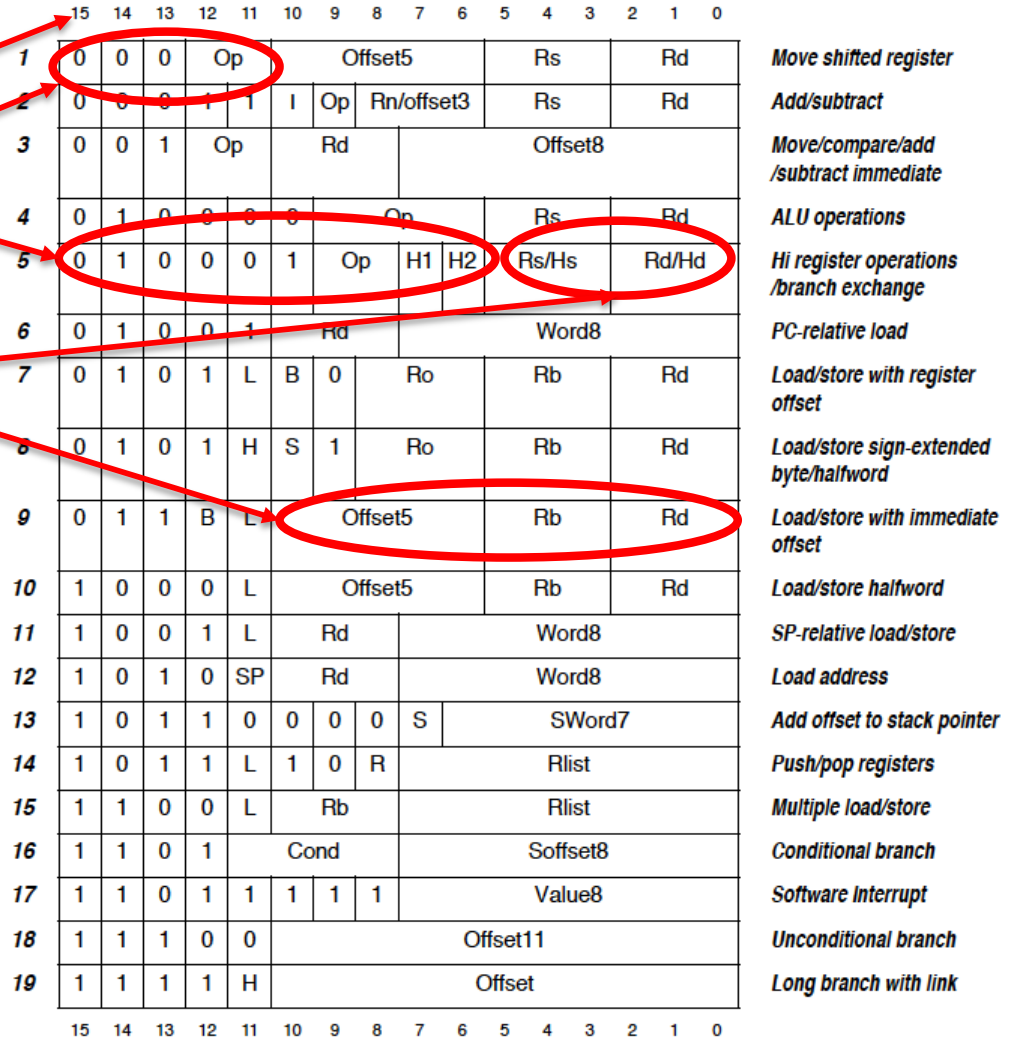
16-bit Instructions

Operation
5 to 10 bits

Operands
6 to 11 bits

Cond: N/Z/C/V

No conditional execution



Total
36
instructions

No {S} instructions

Condition always set by all ALU instructions