COA ASSIGNMENT

Done By,
Group 1
CS - 4A

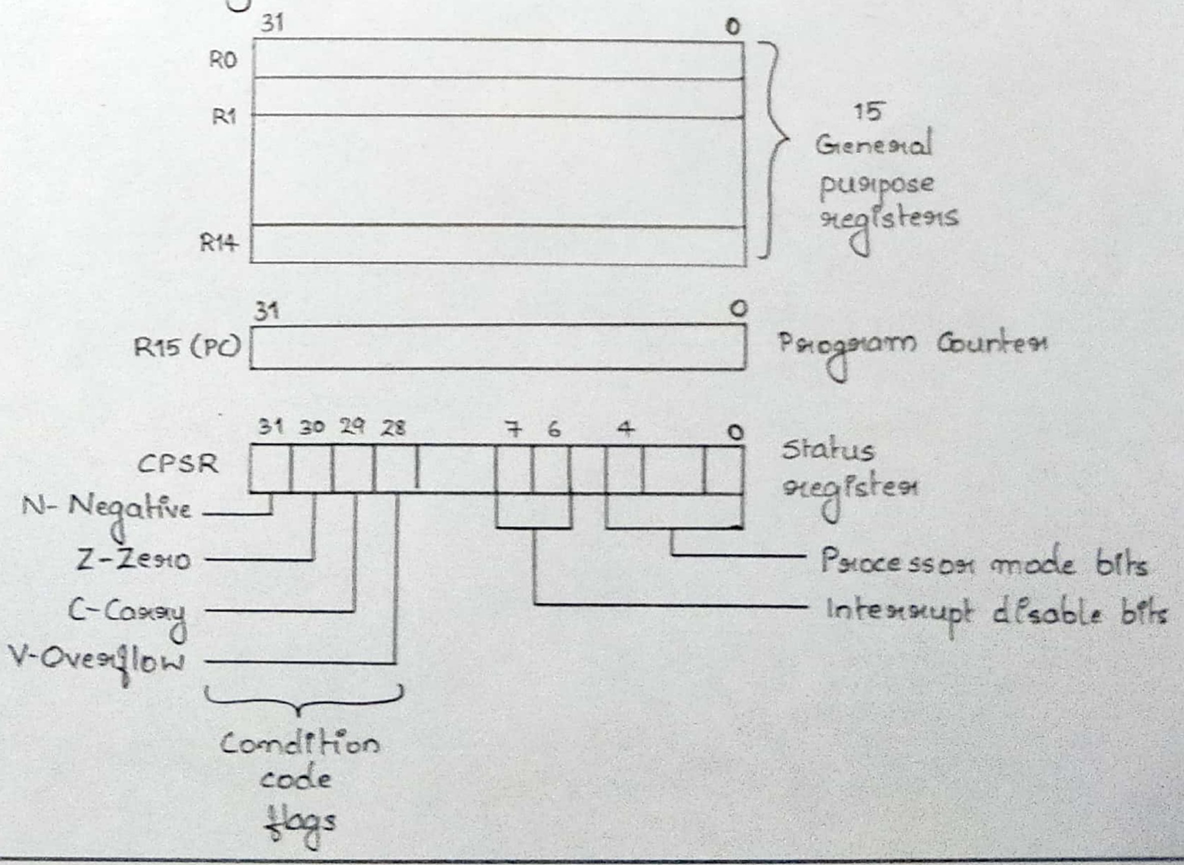
ARM EXAMPLE

REGISTERS, MEMORY ACCESS & DATA TRANSFER

In ARM architectures, memory is byte addressable using 32 bit addresses. Processor registers are also 32 bits long. In moving of data between processor registers & memory, operand length may be 8 bit (byte) or words (32 bit). Both little Endian & big Endian addressing schemes are supported. Memory is accessed only by Load & Store instructions. All arithmetic & logic instructions operate only on data in processor registers. This arrangement is a basic feature of RISC Architectures.

Register Structure

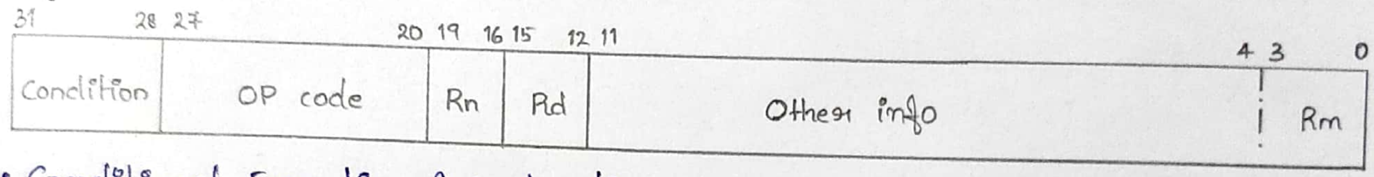
There are 16 32 bit registers labelled R0 to R15. R0 to R14 are general purpose registers & one is dedicated as a program counter (PC). General purpose registers can hold either memory operands or data operands. The Current Program Status Register (CPSR) or simply status register holds the condition code flags, interrupt disable flags & processor mode bits.



There are 15 additional general purpose registers called the banked registers. They are duplicates of some of the R0 to R14 registers. They are used when the processor switches into supervisor mode.

Memory Access Instructions & Addressing Modes

In ARM, access to memory is provided with only Load & Store instructions. The basic encoding format is shown as in the following figure:



• Conditional Execution of Instructions

Unlike others, in ARM processors all instructions are conditionally executed, depending on the condition specified in the instruction. Instruction is executed only when the condition flag is true. Otherwise the processor proceeds to the next instruction. One of the conditions is used to indicate that the instruction is always executed.

• Memory Addressing Modes

For addressing memory operands one of the basic method is generate an EA (Effective Address) of the operand by adding a signed offset to the contents of the base register Rn (which is specified in the instruction). The magnitude of offset may be either an immediate value or the contents of the register Rm.

Examples

LDR Rd, [Rn, #offset]

It performs the operation Rd ← [[Rn] + offset]

LDR Rd, [Rn, Rm]

It performs the operation Rd ← [[Rn] + [Rm]]

If a negative offset is used, Rm must be preceded by a minus sign.
An offset of zero doesn't have to specify explicitly. That is,

LDR Rd, [Rn]

It performs the operation Rd ← [[Rn]]
A byte operand can be moved by using the Opcode LDRB. Similarly Store has the mnemonics STR & STRB.

Example

STR Rd, [Rn]

It performs the operation [Rn] ← [Rd]
Generally we can define 3 addressing modes in ARM processors.

• Pre-Indexed Mode

Effective address of the operand is the sum of contents of base register Rn & an offset value.

• Pre-Indexed with Write Back Mode

It is working in the same way as pre-indexed mode except that effective address is written back to Rn.

• Post-Indexed Mode

The effective address of the operand is the contents of Rn. The offset is then added to this address & the result is written back into Rn.

Register Move Instructions

To copy the contents of register Rm into register Rd, ARM uses the following instruction

MOV Rd, Rm

To load an immediate value in the register Rd, the instruction will be

MOV Rd, #immediate value

Example

MOV R0, #70

Places the value 70 in register R0.

ARITHMETIC INSTRUCTIONS

ARM instruction set has a number of arithmetic & logic operations. The operands may be in general purpose registers or may give as an immediate operand. Memory operands are not allowed in these instructions.

The general format for arithmetic instruction is,

OP code Rd, Rn, Rm

Operation specified by the OP code is performed on operands in general purpose registers Rn & Rm. The result is placed in register Rd.

Example

ADD R0, R2, R4

Adds the content of R2 & R4 & places the sum in register R0.

It performs the operation, R0 ← [R2] + [R4]

ADD R0, R3, #17

Adds the content of R3 & 17 & stores the sum in R0.

It performs the operation, R0 ← [R3] + 17

The immediate value is contained in the 8 bit field on bits b_{7-0} of the instruction. The second operand can be shifted or rotated before being used in the instruction. When a shift or rotation is required, it is specified last in the assembly language expression for the instruction.

Example

ADD R0, R1, R5, LSL #4

The second operand contained in register R5 is shifted left 4 bit positions & it is then added to the contents of register R1 & sum is placed in register R0. Two versions of multiply instructions are there.

1. Multiplies the contents of two registers & places the low order 32 bits of the product in a third register. Higher order bits of the product, if any, are discarded.

MUL R0,R1,R2

It performs the operation, R0 ← [R1] * [R2]

2. Second version called Multiply Accumulate specifies a fourth register whose contents are added to the product before storing the result in the destination register.

MLA R0, R1, R2, R3

It performs the operation, R0 ← [R1] * [R2] + [R3]

This method is often used in numerical algorithms for digital signal processing.