Lab Exercise and Summary

Summary should cover Introduction, Procedure, Data Analysis and Evaluation.

Student's Signature: _____ Date: _____

LABORATORY SKILLS ASSESSMENT (Cognitive)

Obtained Marks	**/10**

Instructor's Signature: _____ Date: _____

LABORATORY SKILLS ASSESSMENT (Psychomotor)

Psychomotor Activity 1: Experimental Setup

S. No	Criteria	Performance	Description	Marks
1	**Keil Assembly Project Setup (4)**	Level 4	Correctly prepares the required Keil assembly project setup for the experiment independently and according to the laboratory requirements.	4
		Level 3	Correctly prepares the required Keil assembly project setup with minor errors or assistance.	3
		Level 2	Partially prepares the required Keil assembly project setup and requires considerable assistance.	2
		Level 1	Shows limited ability to prepare the required Keil assembly project setup and requires substantial assistance.	1
2	**Assembly Program & Data-Area Setup (4)**	Level 4	Correctly prepares the required assembly program and data-area setup for the experiment independently and according to the laboratory requirements.	4
		Level 3	Correctly prepares the required assembly program and data-area setup with minor errors or assistance.	3
		Level 2	Partially prepares the required assembly program and data-area setup and requires considerable assistance.	2
		Level 1	Shows limited ability to prepare the required assembly program and data-area setup and requires substantial assistance.	1
3	**Debug & Memory-Window Parameters (2)**	Level 2	Correctly sets the required debug and memory-window parameters for the experiment.	2
		Level 1	Sets the basic debug and memory-window parameters with minor errors or assistance.	1
Total Obtained				**/10**

If a student is absent:

Criteria	Description	Marks
Absent	Student is absent from the laboratory activity.	**0**

Instructor's Signature: _____ Date: _____

Psychomotor Activity 2: Implementation + Results

S. No	Criteria	Performance	Description	Marks
1	**Assembly Program & Addressing Modes (4)**	Level 4	Correctly implements the assigned assembly program and addressing modes independently while following appropriate laboratory practices.	4
		Level 3	Implements the assigned assembly program and addressing modes with minor errors or assistance and generally follows proper laboratory practices.	3
		Level 2	Partially implements the assigned assembly program and addressing modes and requires considerable assistance or repeated guidance.	2
		Level 1	Implementation of the assigned assembly program and addressing modes is largely incomplete, requires substantial assistance, or shows poor adherence to proper laboratory and safety practices.	1
2	**Execution & Debugging (2)**	Level 2	Correctly performs the required execution and debugging and verifies the expected operation.	2
		Level 1	Performs the basic execution and debugging with minor errors or assistance.	1
3	**Register & Memory Analysis (4)**	Level 4	Obtains correct register and memory analysis and appropriately verifies and interprets the required experimental results.	4
		Level 3	Obtains mostly correct register and memory analysis with minor errors in verification or interpretation.	3
		Level 2	Obtains partially correct register and memory analysis with noticeable errors or omissions.	2
		Level 1	Register and memory analysis is largely incorrect or incomplete and requires substantial assistance.	1
Total Obtained				**/10**

If a student is absent:

Criteria	Description	Marks
Absent	Student is absent from the laboratory activity.	**0**

Instructor's Signature: _____ Date: _____

LABORATORY SKILLS ASSESSMENT (Affective)

Affective Activity: Lab Report + Viva Voce

S. No	Criteria	Performance	Description	Marks
1	**Lab Report Completeness & Organization (4)**	Level 4	Report is complete, well-organized, properly presented, and demonstrates responsible completion of the assigned laboratory work.	4
		Level 3	Report is complete with minor omissions or organizational issues and demonstrates satisfactory responsibility for the assigned work.	3
		Level 2	Report is partially complete with several omissions and demonstrates limited responsibility for the assigned work.	2
		Level 1	Report has major omissions or poor organization and demonstrates minimal responsibility for the assigned work.	1
2	**Register/Memory Output & Conclusion (2)**	Level 2	The register/memory output and conclusion are correctly presented and discussed, demonstrating ownership of the assigned work.	2
		Level 1	The register/memory output and conclusion are presented with minor errors or omissions.	1
3	**Viva Voce (4)**	Level 4	Clearly explains the assigned experiment, assembly directives, addressing modes, program explanation, working, and results, demonstrating individual responsibility, participation, and effective communication.	4
		Level 3	Explains most of the assigned experiment and results correctly, with minor difficulty, and demonstrates good individual participation.	3
		Level 2	Explains basic aspects of the assigned experiment with some prompting and demonstrates basic participation.	2
		Level 1	Demonstrates limited understanding or involvement and requires substantial prompting to explain the assigned work.	1
Total Obtained				**/10**

The student has not submitted the report and has not appeared in the Viva Voce.

Criteria	Description	Marks
Lab Report and Results Not Submitted	Student has not submitted the report and results file.	**0**
Absent in Viva Voce	Student is absent from the viva voce.	**0**

Instructor's Signature: _____ Date: _____

Experiment No 4

ARM assembly programming using Arithmetic Instructions

Objectives

- To learn about Arithmetic, Logical and Shift Instructions in Assembly and how to use them.
- Perform tasks that will use such instructions

ARM Assembly has variety of arithmetic and logical assembly instructions, and the operations are performed in the ALU. ARM arithmetic instructions can perform the following operations:

– Addition, Subtraction
– Multiplication, Division

ARM Logic instructions perform following bit-wise operations:

– AND, OR
– XOR, NOT

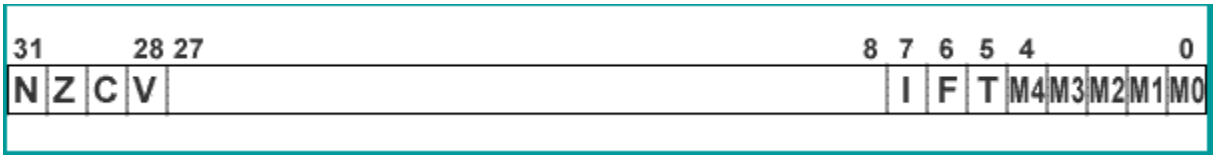
Arithmetic and Logic Instructions affect the following flags in the status register:
– Carry Flag (C)
– Sign Flag (N)
– Zero Flag (Z)
– Overflow Flag (V)

Status Register

The **status register** is a hardware **register** that contains information about the state of the processor. Individual bits are implicitly or explicitly read and/or written by the machine code instructions executing on the processor.

The amount of **registers** depends on the **ARM** version. According to the **ARM** Reference Manual, **there** are 30 general-purpose 32-bit **registers**, with the exception of ARMv6-M and ARMv7-M based processors.

The Current Program Status Register is a 32-bit wide register used in the ARM architecture to record various pieces of information regarding the state of the program being executed by the processor and the state of the processor. This information is recorded by setting or clearing specific bits in the register.



The top four bits (bits 31, 30, 29, and 28) are the condition code (**cc**) bits and are of most interest to us. Condition code bits are sometimes referred to as "flags". The lowest 8 bits (bit 7 through to bit 0) store information about the processor's own state. The remaining bits (i.e. bit 27 to bit 8) are currently unused in most ARM processors.

The N bit is the "negative flag" and indicates that a value is negative.The Z bit is the "zero flag" and is set when an appropriate instruction produces a zero result.The C bit is the "carry flag" but it can also be used to indicate "borrows" (from subtraction operations) and "extends" (from shift instructions (LINK)).

The V bit is the "overflow flag" which is set if an instruction produces a result that overflows and hence may go beyond the range of numbers that can be represented in 2's complement signed format.

	Logical Instruction	**Arithmetic Instruction**
Flag		
Negative (N='1')	No meaning	Bit 31 of the result has been set Indicates a negative number in signed operations
Zero (Z='1')	Result is all zeroes	Result of operation was zero
Carry (C='1')	After Shift operation '1' was left in carry flag	Result was greater than 32 bits
oVerflow (V='1')	No meaning	Result was greater than 31 bits Indicates a possible corruption of the sign bit in signed numbers

For completeness, the other state bits are:

The I and F bits which determine whether interrupts (such as requests for input/output) are enabled or disabled. The T bit which indicates whether the processor is in "Thumb" mode, wherethe processor can execute a subset of the assembly language as 16-bit compact instructions. As Thumb code packs more instructions into the same amount of memory, it is an effective solution to applications where physical memory is at a premium.
The M4 to M0 bits are the mode bits. Application programs normally run in user mode (where the mode bits are 10000). Whenever an interrupt or similar event occurs, the processor switches into one of the alternative modes allowing the software handler greater privileges with regard to memory manipulation.

Bit	Name	Definition
[31]	N	Overflow flag: 1 = overflow in last operation 0 = no overflow.
[30]	Z	Zero flag: 1 = result of 0 in last operation 0 = nonzero result.
[29]	C	Carry/borrow flag: 1 = carry or borrow in last operation 0 = no carry or borrow.
[28]	V	Negative or less than flag: 1 = result negative or less than in last operation 0 = result positive or greater than.

Bit	Name	Definition
[27:8]	-	Reserved
[7]	I	IRQ disable bit: 1 = IRQ interrupts disabled 0 = IRQ interrupts enabled.
[6]	F	FIQ disable bit: 1 = FIQ interrupts disabled 0 = FIQ interrupts enabled.
[5]	T	Thumb state flag: 1 = processor operating in Thumb state 0 = processor operating in ARM state.
[4:0]	M	Mode field: b10000 = User mode b10001 = FIQ mode b10010 = IRQ mode b10011 = Supervisor mode b10111 = Abort mode b11011 = Undefined mode b11111 = System mode.

Writing a value to M [4:0] that is not listed in causes the processor to enter an unrecoverable state. If this occurs, apply Reset.

Registers

The ARM processor has 37 32-bit registers:

- 16 general-purpose registers
- 1 Current Program Status Register
- 15 banked (alternate), mode-specific, general-purpose registers
- 5 banked (alternate), mode-specific Saved Program Status Registers.

These registers are not all accessible at the same time. The processor state and processor

operating mode determine which registers are available to the programmer. Figure 1 shows the register set.

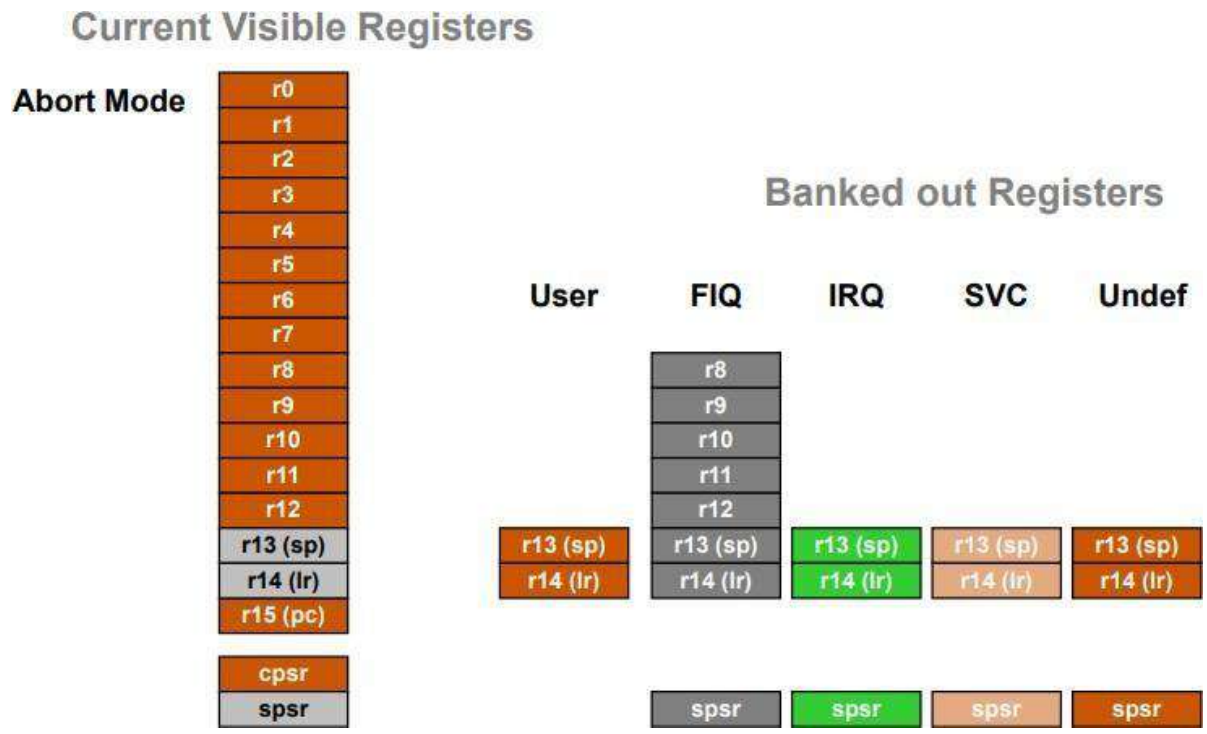


Figure 1 register set

The banked registers are discrete physical registers in the processor that are mapped to the available registers depending on the current processor operating mode. The contents of banked registers are preserved across operating mode changes. Each banked register has a mode identifier to indicate the operating mode. Table below lists the banked register mode identifiers.

Mode	**Identifier**
User[1]	_usr[2]
Fast interrupt	_fiq
Interrupt	_irq
Supervisor	_svc
Abort	_abt
System	_usr
Undefined	_und

[1] User mode and System mode use the same registers.

[2] The _usr identifier is omitted unless it is necessary to distinguish the User or System mode register from another banked register.

The banked r13 and r14 general-purpose registers can be used as mode-specific stack pointers and link registers. For fast interrupt handling, the seven banked general-purpose FIQ mode registers, r8_fiq-r14, can be used to reduce the overhead of saving registers.

The r13, r14, and r15 general-purpose registers also have the following special functions:

Stack pointer

By convention, r13 is used as the Stack Pointer (SP).

Link register

Register r14 is the subroutine Link Register (LR).

The LR receives the return address from r15 when a Branch with Link (BL or BLX) instruction is executed.

At all other times, you can treat r14 as a general-purpose register The banked r14 registers r14_svc, r14_irq, r14_fiq, r14_abt, and r14_und are similarly used to hold the return values when exceptions arise, or when BL or BLX instructions are executed within interrupt or exception routines.

Program counter

Register r15 is the Program Counter (PC).

In ARM state, bits [1:0] of r15 are Undefined and must be ignored. Bits [31:2] contain the program counter value.

In Thumb state, bit 0 is Undefined and must be ignored. Bits [31:1] contain the program counter value.

Arithmetic Instructions

- Operations are:
  - ADD operand1 + operand2 ; Add
  - ADC operand1 + operand2 + carry ; Add with carry
  - SUB operand1 - operand2 ; Subtract
  - SBC operand1 - operand2 + carry -1 ; Subtract with carry
  - RSB operand2 - operand1 ; Reverse subtract
  - RSC operand2 - operand1 + carry - 1 ; Reverse subtract with carry

Multiplication Instructions

- The Basic ARM provides two multiplication instructions.
- Multiply
  - MUL{<cond>}{S} Rd, Rm, Rs ; Rd = Rm * Rs
- Multiply Accumulate - does addition for free
  - MLA{<cond>}{S} Rd, Rm, Rs,Rn ; Rd = (Rm * Rs) + Rn
- Restrictions on use:
  - Rd and Rm cannot be the same register
    - Can be avoided by swapping Rm and Rs around. This works because multiplication is commutative.
  - Cannot use PC.
- These will be picked up by the assembler if overlooked.
- Operands can be considered signed or unsigned
  - Up to user to interpret correctly.

Multiply-Long & Multiply-Accumulate Long

- **Instructions are**
  - MULL which gives RdHi,RdLo:=Rm*Rs
  - MLAL which gives RdHi,RdLo:=(Rm*Rs)+RdHi,RdLo
- **However the full 64 bit of the result now matter (lower precision multiply instructions simply throws top 32bits away)**
  - Need to specify whether operands are signed or unsigned
- **Therefore syntax of new instructions are:**
  - UMULL{<cond>}{S} RdLo,RdHi,Rm,Rs
  - UMLAL{<cond>}{S} RdLo,RdHi,Rm,Rs
  - SMULL{<cond>}{S} RdLo, RdHi, Rm, Rs
  - SMLAL{<cond>}{S} RdLo, RdHi, Rm, Rs
- **Not generated by the compiler.**

Logical & Shift Instructions

- **Operations are:**

  AND operand1 AND operand2
  EOR operand1 EOR operand2
  ORR operand1 OR operand2
  ORN operand1 NOR operand2
  BIC operand1 AND NOT operand2 [ie bit clear]

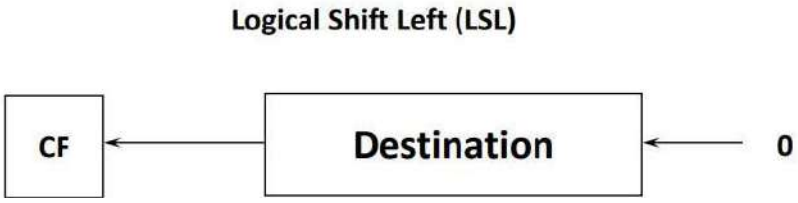
- **Syntax:**
  - <Operation>{<cond>}{S} Rd, Rn, Operand2
- **Examples:**

  AND r0, r1, r2
  BICEQ r2, r3, #7
  EORS r1,r3,r0

Barrel Shifter - Left Shift

- **Shifts left by the specified amount (multiplies by powers of two)**
  e.g.

  LSL #5 => multiply by 32



Barrel Shifter - Right Shifts

Logical Shift Right (LSR)
Shifts right by the specified amount (divides by powers of two) e.g.

LSR #5 = divide by 32



Arithmetic Shift Right (ASR)
Shifts right (divides by powers of two) and preserves the sign bit, for 2's complement operations. e.g.

ASR #5 = divide by 32



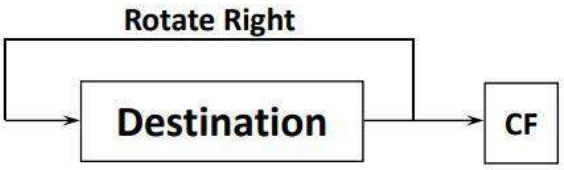
Barrel Shifter - Rotations

Rotate Right (ROR)

Similar to an ASR but the bits wrap around as they leave the LSB and appear as the MSB.

e.g. ROR #5

Note the last bit rotated is also used as the Carry Out.



Rotate Right Extended (RRX)

This operation uses the CPSR C flag as a 33rd bit.

Rotates right by 1 bit. Encoded as ROR #0

