Microprocessor and Assembly Language Lab: Lab 5

May 20, 2025

1 Introduction

This lab is designed to familiarize with different shift instructions for cortex M4 processor and use those to implement boolean operation. Shift operations are sometimes groups with logical operations and sometime they are not. This is because they are not fundamental Boolean operations but they are operations on bits. A shift operation moves all the bits of a word one or more places left or right. Typical shift operations are:

- LSL: Logical shift left
  - Shift the bits left. A 0 enters at the right hand position and the bit in the left hand position is copied into the carry bit.
- LSR: Logical shift right
  - Shift the bits right. A 0 enters at the left hand position and the bit in the right hand position is copied into the carry bit.
- ASR : Arithmetic Shift Right
  - An arithmetic shift right by 0 to 32 places. The vacated bits at the most significant end of the word are filled with zeros if the original value (the source operand) was positive.
- ROR: Rotate right
  - Shift the bits right. The bit shifted out of the right hand position is copied into the left hand position. No bit is lost.

2 ARM Logical Operations

The ARM's logical operations are:

- MVN: MVN r0,r1 : $r0 = \overline{r1}$
- AND: AND r0,r1,r2 : $r0 = r1.r2$
- ORR: OR r0,r1,r2 : $r0 = r1 + r2$
- EOR: XOR r0,r1,r2 : $r0 = r1 \oplus r2$
- BIC: BIC r0,r1,r2 : $r0 = r1.\overline{r2}$

The two unusual instructions are MVN (move negated) and BIC (clear bits). The move negated instruction acts rather like a move instruction (MOV), except that the bits are inverted. Note that the bits in the source register remain unchanged. The BIC instruction clears bits of the first operands when bits of the destination operand are set. This operation is equivalent to an AND between the first and negated second operand. For example, in 8 bits the operation BIC r0,r1,r2 (with r1 = 00001111 and r2 = 11001010) would result in r0 = 11000000. This instruction is sometimes called clear ones corresponding.

3 Your Task

- Write assembly language to perform a simple Boolean operation to calculate the bitwise calculation of $F = W.X + \overline{Y.Z}$
- Suppose we have three words P, Q and R. We are going to apply logical operations to subfields (bit fields) of these registers. We'll use 16-bit arithmetic for simplicity. Suppose that we have three 6-bit bit fields in P, Q, and R as illustrated below. The bit fields are in green and are not in the same position in each word. A bit field is a consecutive sequence of bits that forms a logic entity. Often they are data fields packed in a register, or they may be graphical elements in a display (a row of pixels). However, the following example demonstrates the type of operation you may have to perform on bits.
  P=p15p14 p13 p12 p11 p10 p9 p8 p7 p6 p5 p4 p3 p2 p1 p0 = 0010000011110010
  Q = q15 q14 q13 q12 q11 q10 q9 q8 q7 q6 q5 q4 q3 q2 q1 q0 = 0011000011110000
  R = r15 r14 r13 r12 r11 r10 r9 r8 r7 r6 r5 r4 r3 r2 r1 r0 = 1100010011111000

  Write assembly language to calculate F = (P + Q ⊕ R).111110 using the three 6-bit bit fields.
- Write an assembly language to find the one's complement of a number.
  Sample :
  Input: Number: C123
  Output: Complement: FFFF3EDC
- Write an assembly language program to disassemble a byte into its high and low order nibbles. Divide the least significant byte of the 8-bit variable Value into two 4-bit nibbles and store one nibble in each byte of the 16-bit variable Result. The low-order four bits of the byte will be stored in the low-order four bits of the least significant byte of Result. The high-order four bits of the byte will be stored in the low-order four bits of the most significant byte of Result.
  Sample :
  Input: Number: 5F
  Output: Result: 050F

4 Submission Guideline

1. Your Assembly code with proper comments. (*.s file)
2. A document (*.tex file) that contains:
3. Detail explanation of the code

- Screenshot that shows the state of the system after the code has been loaded.
- Screenshot that shows the situation after the code has been executed.
- Submit as a .zip file. Example: your classroll_lab#.zip (12_lab5.zip)