ARM Assignment-2

Alan Israel (IMT2015526)

*Let us first see what **conditional execution** is.*

Most data processing instructions can optionally update the condition flags in the Application Program Status Register (**APSR**) according to the result of the operation. You can execute an instruction conditionally, based on the condition flags set in another instruction, either:

- immediately after the instruction that updated the flags
- after any number of intervening instructions that have not updated the flags.

Conditional execution is available by using conditional branches or by adding condition code suffixes to instructions. Conditional instructions, except for conditional branches, must be inside an **If-Then** instruction block.

Therefore, based on the two rules mentioned above, we understand that conditional execution requires a preceding IT instruction. An instruction with a condition code is only executed if the condition code flags in the APSR meet the specified condition.

*Let us now see what **IT** condition instruction is.*

The IT instruction makes up to four following instructions conditional. The conditions can be all the same, or some of them can be the logical inverse of the others. The conditional instructions following the IT instruction form the IT block. The instructions in the IT block, including any branches, must specify the condition in the **{cond}** part of their syntax.

So, what are the type of instructions that can be inside the IT block?

Each instruction inside the IT block must specify a condition code suffix that is either the same or logical inverse as for the other instructions in the block. Therefore the 4 errors which were occurring in the first code are due to the MOV instructions which have no condition code.

Does that mean any instruction which has a condition code can be under the IT block?

NO! The primary intent of the Assembler is to gather the instructions after a condition check and assemble them in a manner that they do no alter the APSR flags. Therefore, the assembler will check that the condition you gave to it is consistent with those on the individual instructions. The then conditions must match the condition code, and any else conditions must be the opposite condition.

For example,

```
ITTTE LT
ADDLT r3, r3, r8
MOVLT r4, #0
MOVLT R0, #1
SUBGE r3, r3, r3
```

We can observe that the three instructions immediately after the ITTTE instruction which pertain to the *then* block carry the condition LT which will ensure that the flags aren't altered in case the LT satisfies. In the *else* block we have the instruction which has GE as the condition code which is the logical inverse of the code LT.