Lab-1. EE3401, Jan-Apr2024

You can use the cpu emulator https://cpulator.01xz.net/?sys=arm.

You will be using the instructions we have covered in class (data processing and branching instructions) for the following exercises.

1. You will also need to store data in memory. The website above has total program memory starting from #0000 0000 and going till #FFFF FFFF. See the tab named “Memory”. When you write a program in assembly and compile it, the program’s opcode will be stored in memory starting from #0000 0000.
For example, write this code in the “Editor” tab:

```
.global _start
_start:
    mov r0, #4
```

Press the “Compile and Load” button and it gives you a disassembly in the other tab. Now examine “Address” and “Opcode” columns in disassembly tab, and look at the Memory tab, at the memory location of #0000 0000. (you may need to scroll, since the memory tab wraps the display after #ffff ffff is reached).

You programs will normally take very small space in memory, rest of it is available for data. To store a value of “4” in memory:

```
.global _start
.equ a, 4
.equ b, 0x100
_start:
    ldr r0, =a
    ldr r1, =b
    str r1, [r1]
```

Press the “Compile and Load” button and look at the disassembly and memory contents.
- Ldr and str are load and store register instructions, which we will cover in next class.
- .equ is like #define macro of C, it gets substituted where =var is used. You can use decimal or hexadecimal numbers with it.

Above program stores value “4” in register r0, then value 0x100 (256) in register r1. Then it stores, the value in register r1 (the second argument) to the location pointed by the register r1. You can examine the memory location 0x100 after executing this program.

2. Find the smallest of three given numbers. Store these numbers at location 0x1000 using the method shown above.

1	0x102C2056 0x70409254 0x2344343a

Then the output should be in register r0: 0x102C2056.

3. Find the sum of squares for a series of integers. Their total number n and values are located at a given memory location, e.g.
n=5 and numbers = 0x1 0x2 0x3 0x4 0x5

4. Write code to add two 64-bit numbers stored as in exercise 1 above.

5. Write ARM assembly for the following function:

```
1  while (n--) *to++ = *from++;
```

n=32. It copies data to one memory location (pointed by “to”) from another (pointed by “from”). Use location “to” = 0x1000, “from”= 0x2000.