**CPULator Assembly Lab 1**

In the following lab, students will use the CPUlator found at the following website: [CPUlator Computer System Simulator (01xz.net)](https://cpulator.01xz.net/)

For the CPUlator labs, students are to follow the example instructions, take a snapshot image of their work and upload their work on Canvas.

Follow the below instructions using the CPUlator and screenshot your and upload your results.

* Pick ARMv7 => ARMv7 DE1-SoC
* Each register has 8 zeros. Each zero represents a Hexadecimal
* Hexadecimals have 4 bits, so 8 hexadecimals make 32 bit machine
* Refer to 32 bit as “word” in term of size, so changing a word of data means changing 32 bits of data. Word refers to total amount data stored in a register. If you had 64 bits then word would be 64bit.
* Some registries have special purposes:
  + In general register R0 to R6 used for general purposes
  + R7 used for system calls -talk to operating system. Store numerical value in r7 and that will map to a table for some specific action. For instance if store value 1, then interrupt program, the operating system will read and interpret that as end the program.
  + Sp, lr, and pc are special operators
    - sp relates to stack pointer. It will always tell us the address of next available piece of memory on the stack
    - if click on memory you will see that you have a bunch of stack memory stored in RAM of computer.

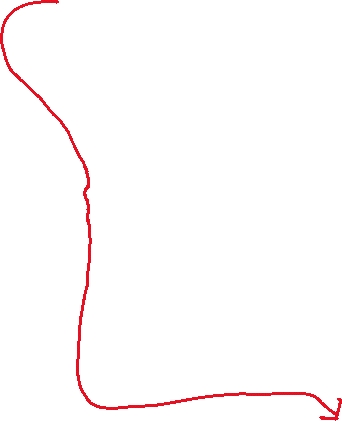
.Graphical user interface, text, application

Description automatically generated

* + - if click on memory you will see that you have a bunch of stack memory stored in RAM of computer.
    - Memory location represented in hex. So 00000000 is zero in hex, 00000010 is 16 in hex.

Graphical user interface, application, table, Excel

Description automatically generated



* + - lr is the link register. High level languages have functions, and functions have returns. Returns allow us to move back to the location of what called the function. The link register stores the location that a function should return to.
    - pc is the program counter, keeps track of location of next instruction to execute. All instructions stored in memory.
    - cpsr – used to store information about our program. If you subtract you might end up with a negative number. Use two complement in binary. The cspr sets a flag in memory that says the result of previous operation was negative, so cpsr says to interpret that registry as negative. The flag tells if result was negative, if it was zero, carry or overflow

compile and run the following program in the simulator:

* MOV R0, #30 – stores constant 30 in register r0. You can store information from stack or other registries

Graphical user interface, text, application

Description automatically generated

* r7,#1 exits the program, and SWI, 0 is software interrupt. It interrupts program and lets operating system take over
* click compile and load – take a snapshot and post for your document submittal
* press step into – what happens. Take a snapshot and include. What is the hex value stored in r0?
* Change format from hexadecimal to decimal.
* Program counter telling us we are at address 4. Look under the disassembly and see the op-code. Look under the memory table at address code. Does the op-code instruction appear at that memory address?

Storing and retrieving data

* Immediate addressing – place constant into register
* Register direct addressing – move what’s in register 0 to registry 1

Graphical user interface, text, application, email

Description automatically generated

Getting data from stack

* To load data from stack, you will have a :data section in your assembly code
* You will declare a label/variable such as list:
* Then a type such as “word” which is 32 bit or “ascii” see Appendix D page 351
* Then you use a LDR to load data from stack into registry. It loads from the stack number or character from the first memory location, then what follows after. “word” tells it 32 bit, if not using word you have specify the length.

Graphical user interface, text, application

Description automatically generated

Change to the memory location and decimal and you can see the numbers stored from our list in memory:

Graphical user interface, application

Description automatically generated

**Adding two numbers:**

* Put 5 in registry 0, and 7 in registry 1
* Add registry 0 and registry 1 and put results in registry 2
* You can ADD, MUL and SUB

Graphical user interface, text, application

Description automatically generated

Adding two numbers continue

* If you are adding a very large number and the result could not be stored in one registry then you need to carry
* The carry flag is set in the “cpsr” has well as if it’s a negative number

Graphical user interface, text, application, email

Description automatically generated