LAB 1
Using Visual Emulator

Download the emulator
https://salmanarif.bitbucket.io/visual/downloads.html

Answer to questions (Q)
Load the following Program in Visual Emulator

; The purpose of this program is to add: aa+bb=cc

; First we define the variables; The variables are stored in locations 0x100-0x108 in the MEMMORY

aa DCD 0x00112233 ;load (32 bit) value 0x00000001 into memory location 0x100 (by default) call it aa
bb DCD 2 ;load (32 bit) value 0x00000002 into memory location 0x104 (by default)
cc DCD 0 ;load (32 bit) value 0x00000000 into memory location 0x108 (by default)

main
LDR r1, =aa ;load memory address of aa into r1 After execution: PC = 0xC+4
LDR r2, [r1] ;load content of memory address in r1 into r2 (that is 0x0001)
LDR r3, =bb
LDR r4, [r3]
ADDS r5, r2, r4 ;r2+r4 -> r5 and update flags
LDR r6, =cc ;load memory address of cc into r6
STR r5, [r6] ;store the sum into variable cc
; this section shows that the sum is in fact loaded into cc
LDR r7, =cc
LDR r8, [r7]
; cc+1---> r9
ADDS r9, r8, #1
END

<table>
<thead>
<tr>
<th>DCB</th>
<th>Define Constant Byte</th>
<th>Reserve 8-bit values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCW</td>
<td>Define Constant Half-word</td>
<td>Reserve 16-bit values</td>
</tr>
<tr>
<td>DCD</td>
<td>Define Constant Word</td>
<td>Reserve 32-bit values</td>
</tr>
<tr>
<td>DCQ</td>
<td>Define Constant Byte</td>
<td>Reserve 64-bit values</td>
</tr>
</tbody>
</table>
Execute the Program:

Click on Step Forward
See what happens as you step through

Note how registers R1-R9 change

Note that the memory address for aa is 0x100, for bb is 0x104 and so far....

Click on HEX BIN and DEC and see the values.

Question:

Q1. What is the largest value of aa variable in HEX and DECIMAL? How many bits are required to represent this value?

Q2. What is the memory address of cc?
See what happens as you step through

Note that every time an instruction is executed, the PC value is increased.

Note that as ADDS is executed, the color of CSPR changes indicating it is being updated.

Question:

Q3. By the time the program is completed, what is the value of PC?
See what happens as you step through

Note that every time an instruction is executed the INSTRUCTION clock cycle is increased.

Question:

Q4. How many instruction cycles are required to execute ADDS instruction?

Q5. How many instruction cycles are required to execute STR instruction?

Q6. How many instruction cycles are required to complete this code?
As you execute ....Click on MEMORY when LDR is executed. Note that you can see the Memory address for variable aa this is where it is saved.

Q7. How many bytes variable aa has?
There are two separate memory structures for DATA and INSTRUCTIONS. The DATA MEMORY in this EMUALTOR starts from 0x100. However, the Instruction MEMORY starts from 0x0. We know that based on the value of the PC.

Note that in this case as PC increases, it points to a new instruction.

Q8. What is the PC value AFTER **ADD**S **r9, r8, #1** is executed?

Q9. What Does PC stand for?
ADD the following code to the end of your program.
Step through the program and see how R1 changes.
Q10. Clearly explain what is happening to R1. What does RRX instruction do?
Q11. Mathematically every time RRX is performed what happens?
Q12. What does B instruction do?

```
R0 0x0
Dec Bin Hex
R1 0x4
Dec Bin Hex
```

```
R2 0x1233
Dec Bin Hex
R3 0x104
Dec Bin Hex
R4 0x2
Dec Bin Hex
```

```
R0 R1 R2 R3 R4
```

```
0 0000000000000000000000000000000000000000000000000000000000000100
```

```
```

```
Reset to continue editing code
The purpose of this program is to add: aa+bb=cc
First we define the variables; The variables are stored in location
DCD 0x00112233
load (32 bit) value 0x00000001 into memory location
DCD 2
load (32 bit) value 0x00000002 into memory location
DCD 0
load (32 bit) value 0x00000000 into memory location
LDR r1, =aa
load memory address of aa into r1 After execution
LDR r2, [r1]
load content of memory address in r1 into
LDR r3, =bb
LDR r4, [r3]
ADDs r5, r2, r4
.r2+r4 -> r5 and update flags
LDR r6, =cc
load memory address of cc into r6
STR r5, [r6]
store the sum into variable cc
LDR r7, =cc
this section shows that the sum is in fact loaded into cc
LDR r8, [r7]
let's check cc+1-->r9
ADDs r9, r8, #1
```

```
RRX R1, R3 ; ROTATE R4 BY ONE BIT AND PLACE IN R1
shiftme
RRX R1, R1
B shiftme
END
```

```
Question:
Q4. How many instruction cycles are required to execute ADDS instruction?
Q5. How many instruction cycles are required to execute STR instruction?
Q6. How many instruction cycles are required to complete this code?
```

```
Click on a line number to restore program to state at that line number.
```

```
Line Number | Value
-------------|-------
1 0x100
19 0x82
22 0x41
22 0x20
22 0x10
22 0x8
22 0x4
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```
Try the following commands. In each case specify the purpose of the command, what happens to the destination register.

<table>
<thead>
<tr>
<th>Examples</th>
<th>Purpose?</th>
<th>What happens to the dest.?</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV r0, #42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOV r2, r3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MVN r1, r0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOV r0, r0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADD r0, r1, r2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUB r5, r3, #10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSB r2, r5, #0xFF00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AND r8, r7, r2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORR r11, r11, #1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIC r11, r11, #1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EOR r11, r11, #1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC r5, r3 r1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MUL r6, r4, r2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Q.13 Practice With the Following Commands

TRY ONE COMMAND AT A TIME:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDS R1, R2, R3</td>
<td>Add R1 and R2 and store the result in R3.</td>
</tr>
<tr>
<td>MOV R1, R3</td>
<td>Move the value from R3 to R1.</td>
</tr>
<tr>
<td>ADC R1,R2,0 ;R9-R9+0+C</td>
<td>Add the value of R2 to R1 with carry from R9.</td>
</tr>
<tr>
<td>MOV R2, #0x4F</td>
<td>Move the value 0x4F to R2.</td>
</tr>
<tr>
<td>MOV R3, #0x39</td>
<td>Move the value 0x39 to R3.</td>
</tr>
<tr>
<td>SUBS R4,R2,R3 ;R4=R2-R3</td>
<td>Subtract R3 from R2 and store the result in R4.</td>
</tr>
<tr>
<td>MOV R2, #0x4F</td>
<td>Move the value 0x4F to R2.</td>
</tr>
<tr>
<td>SUBS R4,R2,#0x05 ;R4=R2-5</td>
<td>Subtract 5 from R4.</td>
</tr>
<tr>
<td>MOV R2, R2, R1</td>
<td>Move the value of R2 to R1.</td>
</tr>
<tr>
<td>SUB R2, R2, #0x2 ; Immediate number operand -&gt; R2=0x2</td>
<td>Subtract 2 from R2.</td>
</tr>
<tr>
<td>SUB R0, #1, R3 ; Note this is an error!</td>
<td>Subtract 1 from R3.</td>
</tr>
<tr>
<td>SBC R2, R2, R1 ; Assume Carry is 1</td>
<td>Subtract R1 from R2 with carry set.</td>
</tr>
<tr>
<td>MOV R1, #0x64 ;R1=0x0100 0000 ; alp-word</td>
<td>Move the value 0x64 to R1.</td>
</tr>
<tr>
<td>MOV R2, #0x4F ;R2=0x1000 1000</td>
<td>Move the value 0x4F to R2.</td>
</tr>
<tr>
<td>MUL R3, R2, R1</td>
<td>Multiply R2 and R1 and store the result in R3.</td>
</tr>
</tbody>
</table>

ADD R0, R2,R1, LSL #1 ; R0 = R2 + (R1 << 1) = 0x84

ROR R1, R1, #4 ;R1 = 0x4 — ROTATE TO RIGHT BY 4 BITS
RRX R1, R4 ; ROTATE R4 BY ONE BIT AND PLACE IN R1=0X3

ADD R0, R2,R1, LSL #1 ; R0 = R2 + (R1 << 1) = 0x84

MOV R1, #0x36
MOV R2, #0x06
MOV R3, R2, ROR R1 ;Rotate to right by value of R1—> R3

http://web.sonoma.edu/users/f/farahman/sonoma/courses/es310/310_arm/res
tources/datasheets/list_of_assembly_commands.pdf
Use the following Table to Complete Each Command – Must be submitted

<table>
<thead>
<tr>
<th>Examples</th>
<th>Purpose?</th>
<th>What happens to the dest. ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV r0, #42</td>
<td>Moving a value into a register</td>
<td>Register R0=42 decimal /</td>
</tr>
</tbody>
</table>
Submit the following:

- Answer all the questions.
- Write a program to do the following:
  - Load R5 with 0x5
  - Load R1 with 0x2
  - R2=R1+R2
  - R2=R1+R2
  - Set R5=0x20
  - Store R2 into location pointed by R5
- Let’s say we add 0x9C and 0xFFFFFFF64. Which flags will be set? Take a snap shot of the simulator. Explain your answer.
- Let’s say we add 0x9C and 0xFFFFFFF69. Which flags will be set? Take a snap shot of the simulator. Explain your answer.

- Write a program to do the following:
  - Load R2 with 0x4
  - Load R3 with 0x2
  - Load R4 with 0x4
  - R5= R2-R3
  - R5=R2-R4
  - What is the state of the flags?
- Define a 32-bit register called Steve and set it to 0x2321. Then move the address of Steve to register R2
- Write an instruction to set bit 4 of R7 only
- Write a simple routine that can add TWO 64-bit values. The results must be in Reg R5 & R6