Question 2.1.1

PRITISH WADHWA

- ARMsim code:

```
P09:
    STMFD   SP!     ,{R4 - R7   ,LR}
    ADDS    R9      ,R5         ,R7
    ADCS    R8      ,R4         ,R6
    LDMFD   SP!     ,{R4 - R7   ,PC}
```

```
1  P09:
2      STMFD   SP!     ,{R4 - R7   ,LR}
3      @ A is represented as R4 R5 where R4 has higher byte and R5 has lower byte
4      @ B is represented as R6 R7 where R6 has higher byte and R7 has lower byte
5      ADDS    R9      ,R5         ,R7 @ Here R5 and R7 are added and stored in R9
6                                      @ Carry bit if generated is stored in CPSR Register
7      ADCS    R8      ,R4         ,R6 @ Here R4 and R6 are added along with the carry bit
8                                      @ generated in the previous step and stored in R8
9      @ A + B is stored in R8 R9, R8 being the higher byte and R9 being the lower byte
10     LDMFD   SP!     ,{R4 - R7   ,PC}
```

Question 2.1.2

PRITISH WADHWA

- ARMsim code:

```
P09:
   STMFD   SP!    ,{R4 - R7   ,LR}
   ADDS    R9     ,R5         ,R7
   ADCS    R8     ,R4         ,R6
   LDMFD   SP!    ,{R4 - R7   ,PC}

P10:
   STMFD   SP!    ,{R2 - R7   ,LR}
   MOV     R0     ,R5
   MOV     R5     ,R4
   MOV     R4     ,R3
   BL      P09
   MOV     R10    ,R9
   MOV     R9     ,R8
   MOV     R5     ,R0
   ADC     R8     ,R2         ,R5
   LDMFD   SP!    ,{R2 - R7   ,PC}
```

```
1  P09:
2      STMFD   SP!     ,{R4 - R7   ,LR}
3      ADDS    R9      ,R5         ,R7
4      ADCS    R8      ,R4         ,R6
5      LDMFD   SP!     ,{R4 - R7   ,PC}
6
7  P10:
8      STMFD   SP!     ,{R2 - R7   ,LR}
9      @ A is represented as R2 R3 R4 where R2 is the MSB and R4 is the LSB
10     @ B is represented as R5 R6 R7 where R5 is the MSB and R7 is the LSB
11     MOV     R0      ,R5
12     MOV     R5      ,R4
13     MOV     R4      ,R3
14     BL      P09
15     MOV     R10     ,R9
16     MOV     R9      ,R8
17     MOV     R5      ,R0
18     ADC     R8      ,R2         ,R5
19     @ A + B is stored in R8 R9 R10, R8 being the MSB and R10 being the LSB
20     LDMFD   SP!     ,{R2 - R7   ,PC}
```

Question 2.1.3

PRITISH WADHWA

- ARMsim code:

```
P09:
   STMFD  SP!    ,{R4 - R7   ,LR}
   ADDS   R9     ,R5         ,R7
   ADCS   R8     ,R4         ,R6
   LDMFD  SP!    ,{R4 - R7   ,PC}

P10:
   STMFD  SP!    ,{R2 - R7   ,LR}
   MOV    R0     ,R5
   MOV    R5     ,R4
   MOV    R4     ,R3
   BL     P09
   MOV    R10    ,R9
   MOV    R9     ,R8
   MOV    R5     ,R0
   ADC    R8     ,R2         ,R5
   LDMFD  SP!    ,{R2 - R7   ,PC}

MAIN:
   MOV    R2     ,#0
   MOV    R3     ,#4294967295
   MOV    R4     ,#4294967295
   MOV    R5     ,#0
   MOV    R6     ,#0
   MOV    R7     ,#1
   BL     P10
```

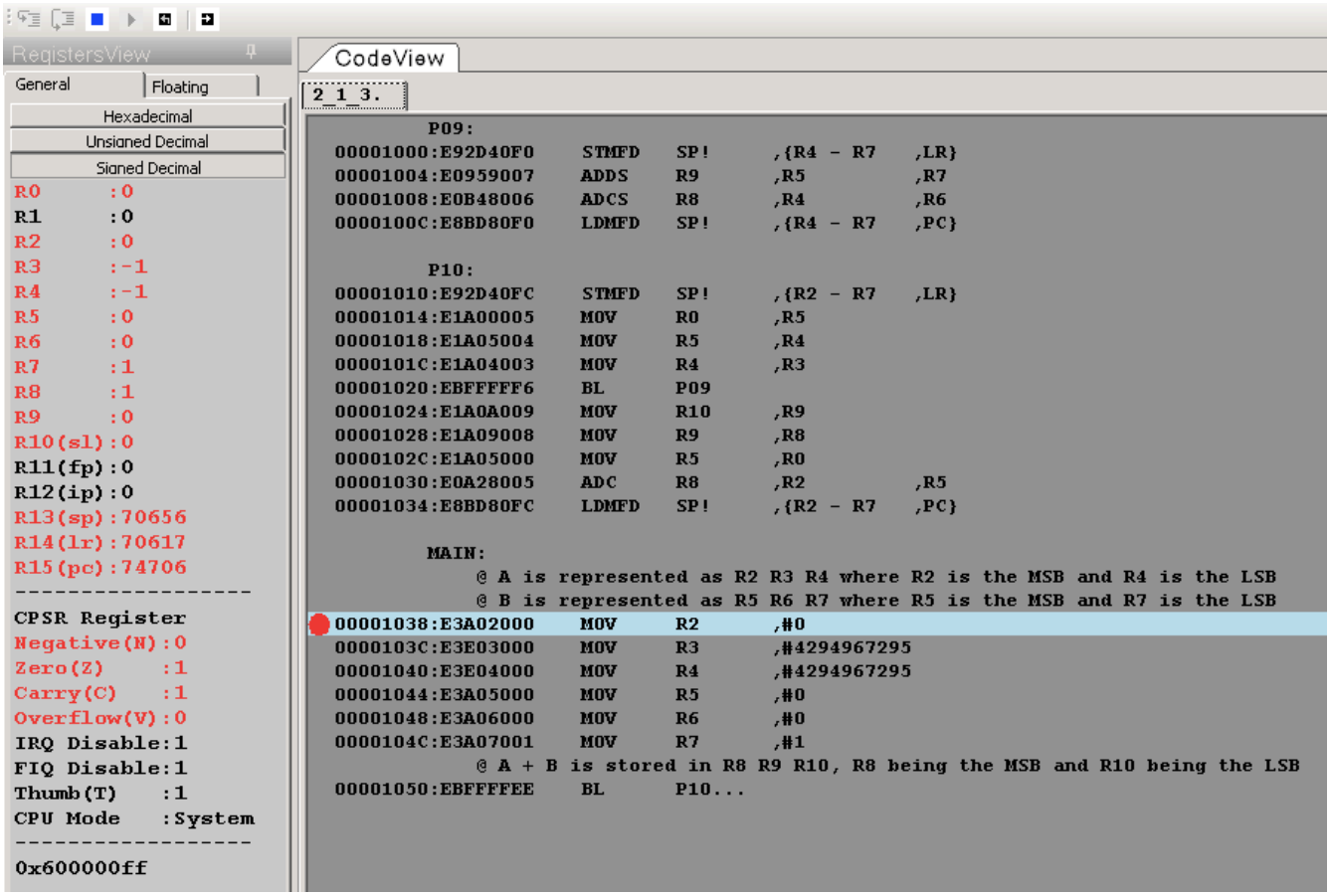
```
P09:
    STMFD   SP!     ,{R4 - R7   ,LR}
    ADDS    R9      ,R5         ,R7
    ADCS    R8      ,R4         ,R6
    LDMFD   SP!     ,{R4 - R7   ,PC}

P10:
    STMFD   SP!     ,{R2 - R7   ,LR}
    MOV     R0      ,R5
    MOV     R5      ,R4
    MOV     R4      ,R3
    BL      P09
    MOV     R10     ,R9
    MOV     R9      ,R8
    MOV     R5      ,R0
    ADC     R8      ,R2         ,R5
    LDMFD   SP!     ,{R2 - R7   ,PC}

MAIN:
    @ A is represented as R2 R3 R4 where R2 is the MSB and R4 is the LSB
    @ B is represented as R5 R6 R7 where R5 is the MSB and R7 is the LSB
    MOV     R2      ,#0
    MOV     R3      ,#4294967295
    MOV     R4      ,#4294967295
```

```
MAIN:
    @ A is represented as R2 R3 R4 where R2 is the MSB and R4 is the LSB
    @ B is represented as R5 R6 R7 where R5 is the MSB and R7 is the LSB
    MOV     R2      ,#0
    MOV     R3      ,#4294967295
    MOV     R4      ,#4294967295
    MOV     R5      ,#0
    MOV     R6      ,#0
    MOV     R7      ,#1
    @ A + B is stored in R8 R9 R10, R8 being the MSB and R10 being the LSB
    BL      P10
```

- ARMsim window:



Question 2.2.1

PRITISH WADHWA

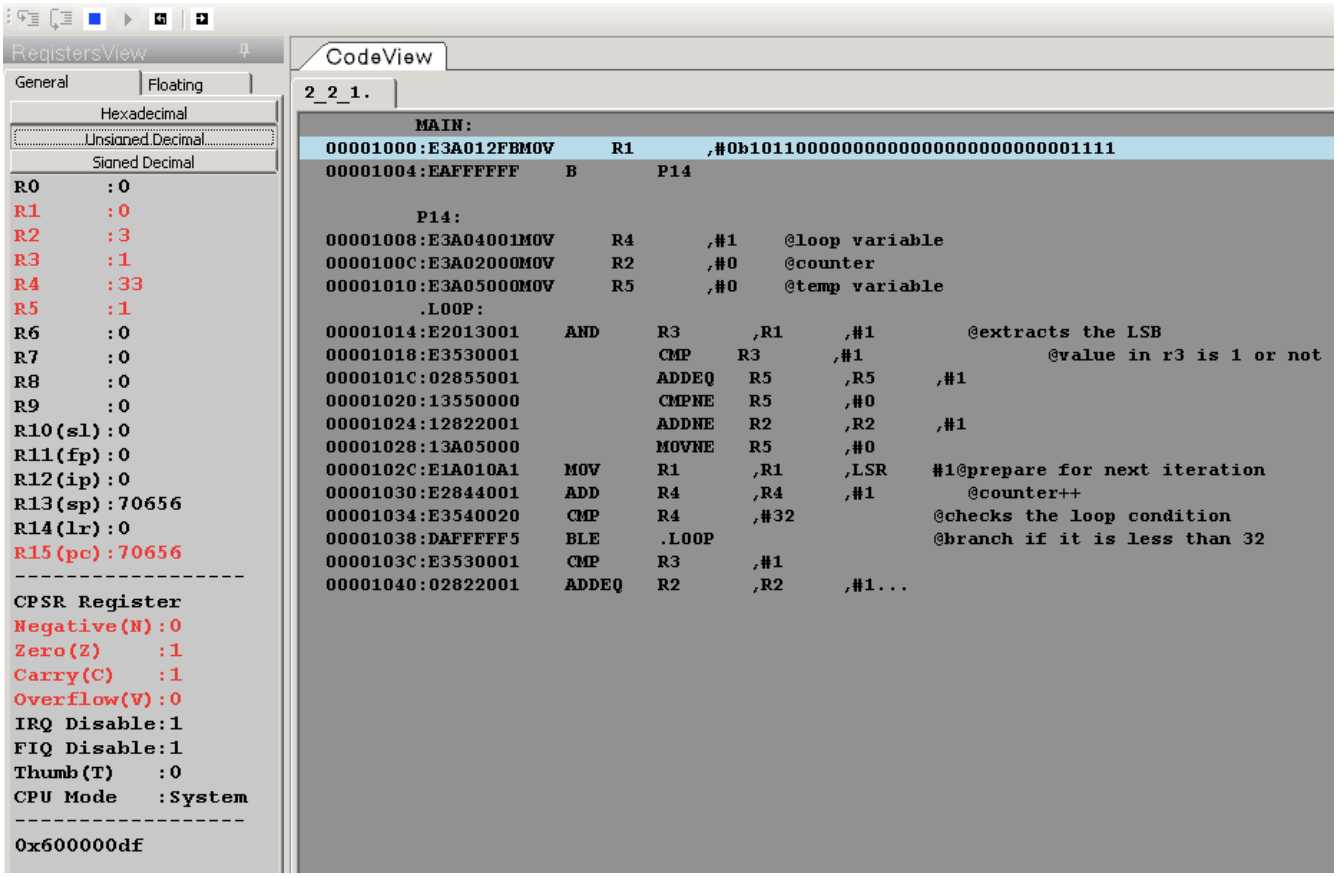
- ARMsim code:

```
MAIN:
   MOV    R1    ,#0b10110000000000000000000000001111
   B      P14

P14:
   MOV    R4    ,#1
   MOV    R2    ,#0
   MOV    R5    ,#0
   .LOOP:
        AND     R3    ,R1    ,#1
        CMP     R3    ,#1
        ADDEQ   R5    ,R5    ,#1
        CMPNE   R5    ,#0
        ADDNE   R2    ,R2    ,#1
        MOVNE   R5    ,#0
        MOV     R1    ,R1    ,LSR   #1
        ADD     R4    ,R4    ,#1
        CMP     R4    ,#32
        BLE     .LOOP
   CMP     R3    ,#1
   ADDEQ   R2    ,R2    ,#1
```

```
1  MAIN:
2      MOV     R1      ,#0b10110000000000000000000000001111
3      B       P14
4
5  P14:
6      MOV     R4      ,#1     @loop variable
7      MOV     R2      ,#0     @counter
8      MOV     R5      ,#0     @temp variable
9      .LOOP:
10             AND     R3      ,R1     ,#1         @extracts the LSB
11             CMP     R3      ,#1                 @value in r3 is 1 or not
12             ADDEQ   R5      ,R5     ,#1
13             CMPNE   R5      ,#0
14             ADDNE   R2      ,R2     ,#1
15             MOVNE   R5      ,#0
16             MOV     R1      ,R1     ,LSR    #1  @prepare for next iteration
17             ADD     R4      ,R4     ,#1         @counter++
18             CMP     R4      ,#32                @checks the loop condition
19             BLE     .LOOP                       @branch if it is less than 32
20     CMP     R3      ,#1
21     ADDEQ   R2      ,R2     ,#1
```

ARMsim window:



Question 2.2.2

PRITISH WADHWA

- The question asked me to use conditional statements as much as possible. I tried my best to do so :-

- My code will witness conditional statements a total of 161 times in the runtime.
- In the loop there are 5 different conditional statements :
    - ADDEQ
        - It will add only when the ‘Z’ flag in CPSR register is set to ‘1’ thanks to the previous compare statement.
    - CMPNE
        - It will compare only when the ‘Z’ flag in CPSR register is set to ‘0’ thanks to the previous compare statement.
    - ADDNE
        - It will add only when the ‘Z’ flag in CPSR register is set to ‘0’ thanks to the previous compare statement.
    - MOVNE
        - It will move only when the ‘Z’ flag in CPSR register is set to ‘0’ thanks to the previous compare statement.
    - BLE
        - It will branch only when the ‘N’ flag in CPSR register is set to ‘1’ thanks to the previous compare statement.

- The above statement are in a loop which will run 32 times each, thus making a total of 160 times.

- Apart from them I have added one more ADDEQ statement in the end which will be executed once thus taking the total count to 161.

Question 2.4 (EMULATOR - 1 - A)

PRITISH WADHWA

CODE :

```
MAIN:
   MOV     R2      ,#0
   MOV     R3      ,#4294967295
   MOV     R4      ,#4294967295
   MOV     R5      ,#0
   MOV     R6      ,#0
   MOV     R7      ,#1
   B       P10

P09:
   ADDS    R9      ,R5       ,R7
   ADCS    R8      ,R4       ,R6
   B       P10HELP

P10:
   MOV     R0      ,R5
   MOV     R5      ,R4
   MOV     R4      ,R3
   B       P09

P10HELP:
   MOV     R10     ,R9
   MOV     R9      ,R8
   MOV     R5      ,R0
   ADC     R8      ,R2       ,R5
```

Open | Run | 250 | Step | Reset

```
1  MAIN:
2       MOV     R2      ,#0
3       MOV     R3      ,#4294967295
4       MOV     R4      ,#4294967295
5       MOV     R5      ,#0
6       MOV     R6      ,#0
7       MOV     R7      ,#1
8       B       P10
9
10 P09:
11      ADDS    R9      ,R5         ,R7
12      ADCS    R8      ,R4         ,R6
13      B       P10HELP
14
15 P10:
16      MOV     R0      ,R5
17      MOV     R5      ,R4
18      MOV     R4      ,R3
19      B       P09
20 P10HELP:
21      MOV     R10     ,R9
22      MOV     R9      ,R8
23      MOV     R5      ,R0
24      ADC     R8      ,R2         ,R5
```

Register	Value
R0	0
R1	0
R2	0
R3	-1
R4	-1
R5	0
R6	0
R7	1
R8	1
R9	0
R10	0
R11	0
R12	0
SP	10000
LR	0
PC	10094
CPSR	60000013

Question 2.4 (EMULATOR – 1 - B)

PRITISH WADHWA

- Code :

```
MAIN:
        MOV     R1      ,#0b10110000000000000000000000001111
        B       P14

P14:
        MOV     R4      ,#1
        MOV     R2      ,#0
        MOV     R5      ,#0
        .LOOP:
                AND     R3      ,R1     ,#1
                CMP     R3      ,#1
                ADDEQ   R5      ,R5     ,#1
                CMPNE   R5      ,#0
                ADDNE   R2      ,R2     ,#1
                MOVNE   R5      ,#0
                MOV     R1      ,R1     ,LSR    #1
                ADD     R4      ,R4     ,#1
                CMP     R4      ,#32
                BLE     .LOOP
        CMP     R3      ,#1
        ADDEQ   R2      ,R2     ,#1
```

Open | Run | 1 | Step | Reset

```
1  MAIN:
2      MOV     R1      ,#0b10110000000000000000000000001111
3      B       P14
4
5  P14:
6      MOV     R4      ,#1
7      MOV     R2      ,#0
8      MOV     R5      ,#0
9      .LOOP:
10             AND     R3      ,R1     ,#1
11             CMP     R3      ,#1
12             ADDEQ   R5      ,R5     ,#1
13             CMPNE   R5      ,#0
14             ADDNE   R2      ,R2     ,#1
15             MOVNE   R5      ,#0
16             MOV     R1      ,R1     ,LSR    #1
17             ADD     R4      ,R4     ,#1
18             CMP     R4      ,#32
19             BLE     .LOOP
20     CMP     R3      ,#1
21     ADDEQ   R2      ,R2     ,#1
```

Register	Value
R0	0
R1	0
R2	3
R3	1
R4	21
R5	1
R6	0
R7	0
R8	0
R9	0
R10	0
R11	0
R12	0
SP	10000
LR	0
PC	100c4
CPSR	60000013

Question 2.4 (EMULATOR - 2 – A)

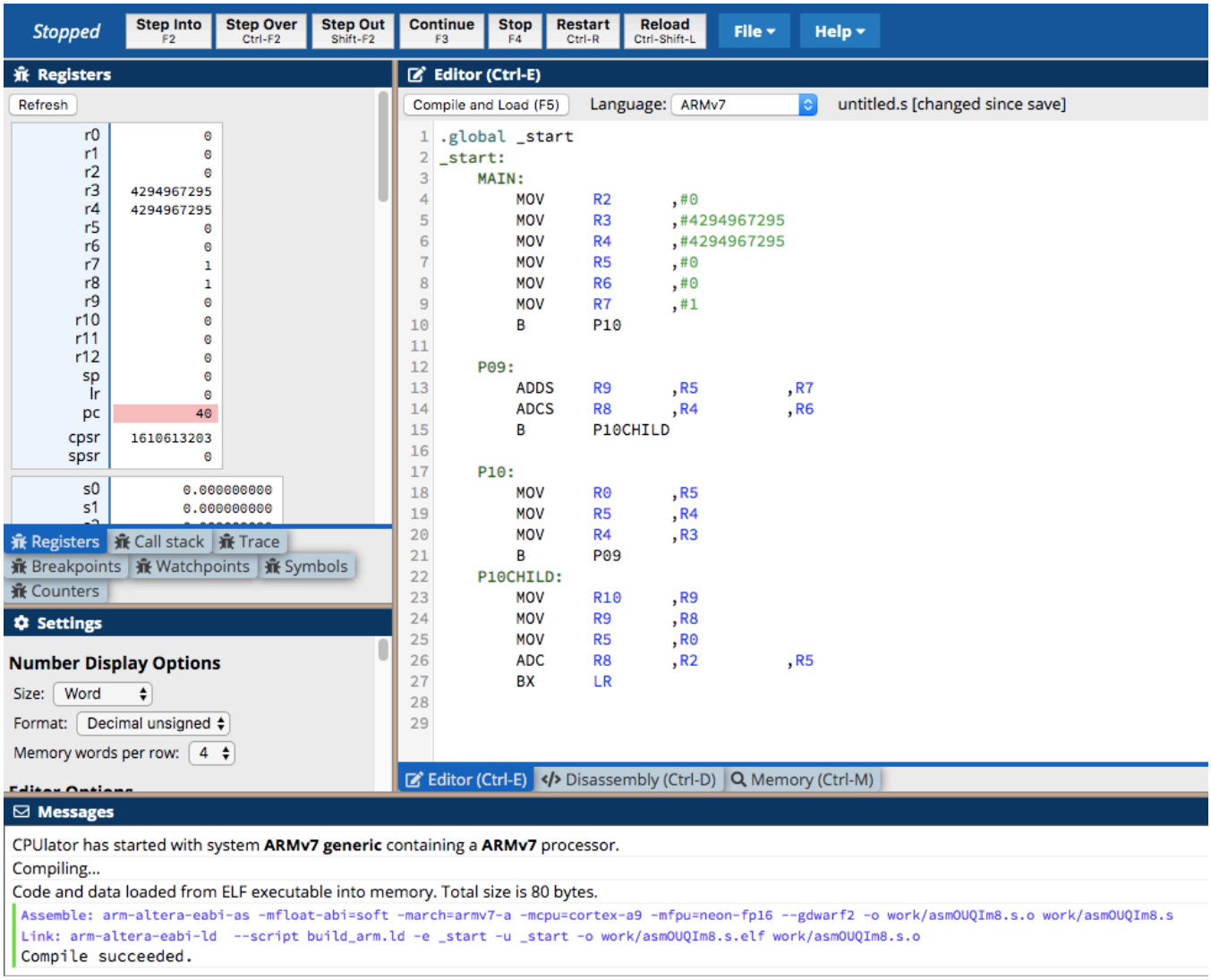
PRITISH WADHWA

- Code :

```
.global _start
_start:
        MAIN:
                MOV     R2      ,#0
                MOV     R3      ,#4294967295
                MOV     R4      ,#4294967295
                MOV     R5      ,#0
                MOV     R6      ,#0
                MOV     R7      ,#1
                B       P10

        P09:
                ADDS    R9      ,R5       ,R7
                ADCS    R8      ,R4       ,R6
                B       P10CHILD

        P10:
                MOV     R0      ,R5
                MOV     R5      ,R4
                MOV     R4      ,R3
                B       P09
        P10CHILD:
                MOV     R10     ,R9
                MOV     R9      ,R8
                MOV     R5      ,R0
                ADC     R8      ,R2       ,R5
                BX      LR
```


Stopped
Step Into F2
Step Over Ctrl-F2
Step Out Shift-F2
Continue F3
Stop F4
Restart Ctrl-R
Reload Ctrl-Shift-L
File
Help
Registers
Refresh
r0 0
r1 0
r2 0
r3 4294967295
r4 4294967295
r5 0
r6 0
r7 1
r8 1
r9 0
r10 0
r11 0
r12 0
sp 0
lr 0
pc 40
cpsr 1610613203
spsr 0
s0 0.000000000
s1 0.000000000
Registers
Call stack
Trace
Breakpoints
Watchpoints
Symbols
Counters
Settings
Number Display Options
Size: Word
Format: Decimal unsigned
Memory words per row: 4
Editor (Ctrl-E)
Compile and Load (F5)
Language: ARMv7
untitled.s [changed since save]
1 .global _start
2 _start:
3 MAIN:
4 MOV R2 ,#0
5 MOV R3 ,#4294967295
6 MOV R4 ,#4294967295
7 MOV R5 ,#0
8 MOV R6 ,#0
9 MOV R7 ,#1
10 B P10
11
12 P09:
13 ADDS R9 ,R5 ,R7
14 ADCS R8 ,R4 ,R6
15 B P10CHILD
16
17 P10:
18 MOV R0 ,R5
19 MOV R5 ,R4
20 MOV R4 ,R3
21 B P09
22 P10CHILD:
23 MOV R10 ,R9
24 MOV R9 ,R8
25 MOV R5 ,R0
26 ADC R8 ,R2 ,R5
27 BX LR
28
29
Editor (Ctrl-E)
Disassembly (Ctrl-D)
Memory (Ctrl-M)
Messages
CPUlator has started with system ARMv7 generic containing a ARMv7 processor.
Compiling...
Code and data loaded from ELF executable into memory. Total size is 80 bytes.
Assemble: arm-altera-eabi-as -mfloat-abi=soft -march=armv7-a -mcpu=cortex-a9 -mfpu=neon-fp16 --gdwarf2 -o work/asmOUQIm8.s.o work/asmOUQIm8.s
Link: arm-altera-eabi-ld --script build_arm.ld -e _start -u _start -o work/asmOUQIm8.s.elf work/asmOUQIm8.s.o
Compile succeeded.

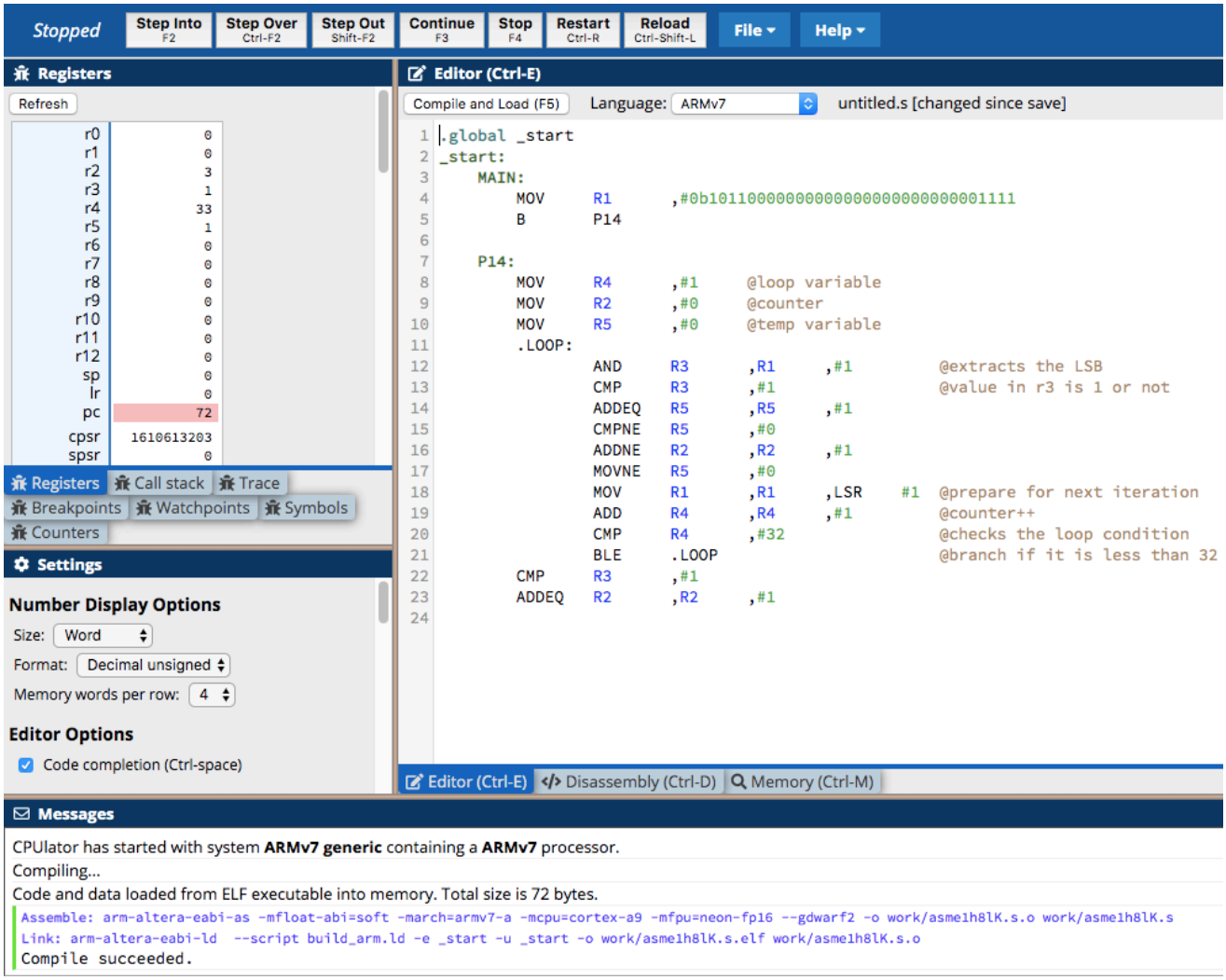
Question 2.4 (EMULATOR - 2 – B)

PRITISH WADHWA

- Code :

```
.global _start
_start:
        MAIN:
                MOV     R1      ,#0b10110000000000000000000000001111
                B       P14

        P14:
                MOV     R4      ,#1
                MOV     R2      ,#0
                MOV     R5      ,#0
                .LOOP:
                        AND     R3      ,R1     ,#1
                        CMP     R3      ,#1
                        ADDEQ   R5      ,R5     ,#1
                        CMPNE   R5      ,#0
                        ADDNE   R2      ,R2     ,#1
                        MOVNE   R5      ,#0
                        MOV     R1      ,R1     ,LSR    #1
                        ADD     R4      ,R4     ,#1
                        CMP     R4      ,#32
                        BLE     .LOOP
                CMP     R3      ,#1
                ADDEQ   R2      ,R2     ,#1
```


Stopped
Step Into F2
Step Over Ctrl-F2
Step Out Shift-F2
Continue F3
Stop F4
Restart Ctrl-R
Reload Ctrl-Shift-L
File
Help
Registers
Refresh
r0 0
r1 0
r2 3
r3 1
r4 33
r5 1
r6 0
r7 0
r8 0
r9 0
r10 0
r11 0
r12 0
sp 0
lr 0
pc 72
cpsr 1610613203
spsr 0
Registers
Call stack
Trace
Breakpoints
Watchpoints
Symbols
Counters
Settings
Number Display Options
Size: Word
Format: Decimal unsigned
Memory words per row: 4
Editor Options
Code completion (Ctrl-space)
Editor (Ctrl-E)
Compile and Load (F5)
Language: ARMv7
untitled.s [changed since save]
1 .global _start
2 _start:
3 MAIN:
4 MOV R1 ,#0b10110000000000000000000000001111
5 B P14
6
7 P14:
8 MOV R4 ,#1 @loop variable
9 MOV R2 ,#0 @counter
10 MOV R5 ,#0 @temp variable
11 .LOOP:
12 AND R3 ,R1 ,#1 @extracts the LSB
13 CMP R3 ,#1 @value in r3 is 1 or not
14 ADDEQ R5 ,R5 ,#1
15 CMPNE R5 ,#0
16 ADDNE R2 ,R2 ,#1
17 MOVNE R5 ,#0
18 MOV R1 ,R1 ,LSR #1 @prepare for next iteration
19 ADD R4 ,R4 ,#1 @counter++
20 CMP R4 ,#32 @checks the loop condition
21 BLE .LOOP @branch if it is less than 32
22 CMP R3 ,#1
23 ADDEQ R2 ,R2 ,#1
24
Editor (Ctrl-E)
Disassembly (Ctrl-D)
Memory (Ctrl-M)
Messages
CPUlator has started with system ARMv7 generic containing a ARMv7 processor.
Compiling...
Code and data loaded from ELF executable into memory. Total size is 72 bytes.
Assemble: arm-altera-eabi-as -mfloat-abi=soft -march=armv7-a -mcpu=cortex-a9 -mfpu=neon-fp16 --gdwarf2 -o work/asme1h8lK.s.o work/asme1h8lK.s
Link: arm-altera-eabi-ld --script build_arm.ld -e _start -u _start -o work/asme1h8lK.s.elf work/asme1h8lK.s.o
Compile succeeded.