

Name: Abu Butt

Professor: Izidor Gertner

Class: CSC 43200

Date: July 8, 2017

Topic: Report on ARM Emulator

Introduction:

ARM stands for originally Acorn RISC Machine, later Advanced RISC Machine, is a family of reduced instruction set computing architectures for computer processors, configured for various environments. Main features of the ARM Instruction Set are all instructions are 32 bits long. Most instructions are executed in a single cycle. Most instructions can be conditionally executed. It is a load/store architecture. It has 3 operands format. In ARM instruction sets there are 16 registers and each register is a 32-bit number. Registers are used to store data.

ARM Emulator (VisUAL) Examples:

Adding Numbers:

Code in C:

```
int total;  
int i;  
  
total = 0;  
for (i = 10; i > 0; i--) {  
    total += i;  
}
```

Above is the code for adding numbers from 1 to 10. This code is written in C. The sum of all numbers from 1 to 10 is 55 in decimal.

ARM ISA:

Example 1: Adding Numbers

The screenshot displays an ARM assembly simulator. On the left, a code editor shows five instructions: `MOV R0, #0`, `MOV R1, #10`, `ADD R0, R0, R1`, `SUBS R1, R1, #1`, and `BNE again`. The instruction `MOV R1, #10` is highlighted. On the right, a register table lists registers R0 through R13, LR, and PC. R0 contains `0x0` and R1 contains `0xA`. R13 contains `0xFF000000`. The PC contains `0xC`. Each register entry has buttons for 'Dec', 'Bin', and 'Hex' views. At the bottom, a status bar shows 'Clock Cycles' (0), 'Current Instruction: 1', 'Total: 2', and 'CSPR Status Bits (NZCV)' (0000).

Register	Value	Dec	Bin	Hex
R0	0x0			
R1	0xA			
R2	0x0			
R3	0x0			
R4	0x0			
R5	0x0			
R6	0x0			
R7	0x0			
R8	0x0			
R9	0x0			
R10	0x0			
R11	0x0			
R12	0x0			
R13	0xFF000000			
LR	0x0			
PC	0xC			

Clock Cycles: 0 Current Instruction: 1 Total: 2
CSPR Status Bits (NZCV): 0 0 0 0

This is the ARM ISA. On the left side of the screen are the ARM instructions for adding numbers from 1 to 10. The right side has all the register that are being used in this problem. This is the beginning of the code. So, you can see the registers R0 and R1 are initialized with their values. The values are in hexadecimal. It also has register PC, which is a program counter,

which is responsible for keeping the instructions that will be executed next.

The screenshot displays an ARM emulator interface. On the left, a code editor shows assembly instructions with line numbers 1 through 7. Line 3, 'again ADD R0, R0, R1', is highlighted. The instructions are: 1 MOV R0, #0; 2 MOV R1, #10; 3 again ADD R0, R0, R1; 4 SUBS R1, R1, #1; 5 BNE again; 6; 7. A 'Reset to continue editing code' button is at the top of the editor. On the right, a register window lists R0 through R13, LR, and PC. R0 and R1 are highlighted in yellow and show the value 0xA. R2 through R13, LR, and PC show 0x0. Each register has buttons for 'Dec', 'Bin', and 'Hex' views. At the bottom, a status bar shows 'Clock Cycles' (0), 'Current Instruction: 1 Total: 3', and 'CSPR Status Bits (NZCV)' (0, 0, 0, 0).

Register	Value	Dec	Bin	Hex
R0	0xA			
R1	0xA			
R2	0x0			
R3	0x0			
R4	0x0			
R5	0x0			
R6	0x0			
R7	0x0			
R8	0x0			
R9	0x0			
R10	0x0			
R11	0x0			
R12	0x0			
R13	0xFF000000			
LR	0x0			
PC	0x10			

Clock Cycles: 0 Current Instruction: 1 Total: 3
CSPR Status Bits (NZCV): 0 0 0 0

This picture is for the loop that is executed in the code. As you can see the again statement in the ARM instructions which mean keep executing the statements until the both values are equal.

This loop will stop executing when the branches (values) are equal. In the loop, we are adding the values of registers R0 and R, and storing it in the register R0. Then we decrement the value of register R1 by 1. This execution can be seen in the right side of the emulator.

Reset to continue editing code

```

1      MOV    R0, #0
2      MOV    R1, #10
3 again ADD    R0, R0, R1
4      SUBS   R1, R1, #1
5      BNE    again
6
7

```

R0	0x37	Dec	Bin	Hex
R1	0x0	Dec	Bin	Hex
R2	0x0	Dec	Bin	Hex
R3	0x0	Dec	Bin	Hex
R4	0x0	Dec	Bin	Hex
R5	0x0	Dec	Bin	Hex
R6	0x0	Dec	Bin	Hex
R7	0x0	Dec	Bin	Hex
R8	0x0	Dec	Bin	Hex
R9	0x0	Dec	Bin	Hex
R10	0x0	Dec	Bin	Hex
R11	0x0	Dec	Bin	Hex
R12	0x0	Dec	Bin	Hex
R13	0xFF000000	Dec	Bin	Hex
LR	0x0	Dec	Bin	Hex
PC	0x18	Dec	Bin	Hex

Clock Cycles
Current Instruction: 1 Total: 50

CSPR Status Bits (NZCV)

0
1
1
0

This is the end of the execution, where register R0 has the value of 37 on hexadecimal, which in decimal is 55. Register R1 is 0. The execution stopped when both branches (values) were equal.

Example 2: Hailstone Sequence3

```

iters ← 0
while n ≠ 1:
    iters ← iters + 1
    if n is odd:
        n ← 3 · n + 1
    else:
        n ← n / 2

```

This is the code in C. It means we are given an integer *n*, we repeatedly apply the top procedure.

If *n* is odd we execute this line of code $n \leftarrow 3 \cdot n + 1$, otherwise $n \leftarrow n / 2$.

ARM ISA:

Example 2: Hailstone Sequence

The screenshot displays an ARM assembly simulator interface. On the left, a list of assembly instructions is shown with line numbers 1 through 13. The instructions are as follows:

```
1 MOV R0, #5 ; R0 is current number
2 MOV R1, #0 ; R1 is count of number of iterations
3 again ADD R1, R1, #1 ; increment number of iterations
4 ANDS R0, R0, #1 ; test whether R0 is odd
5 BEQ even
6 ADD R0, R0, R0, LSL #1 ; if odd, set R0 = R0 * 2
7 ADD R0, R0, #1 ; and repeat (guaranteed R0 > 1)
8 B again
9 even MOV R0, R0, ASR #1 ; if even, set R0 = R0 / 2
10 SUBS R7, R0, #1 ; and repeat if R0 != 1
11 BNE again
12 halt B halt ; infinite loop to stop computation
13
```

On the right side, a table shows the current values of the registers:

Register	Value	Dec	Bin	Hex
R0	0x5			
R1	0x0			
R2	0x0			
R3	0x0			
R4	0x0			
R5	0x0			
R6	0x0			
R7	0x0			
R8	0x0			
R9	0x0			
R10	0x0			
R11	0x0			
R12	0x0			
R13	0xFF000000			
LR	0x0			
PC	0xC			

At the bottom of the simulator, there is a status bar showing "Clock Cycles" and "Current Instruction: 1 Total: 2". Below this, a row of four boxes labeled "CSPR Status Bits (NZCV)" all contain the value "0".

This is the ARM ISA. On the left side of the screen are the ARM instructions for Hailstone Sequence. The right side has all the register that are being used in this problem. This is the beginning of the code. So, you can see the registers R0 and R1 are initialized with their values. The values are in hexadecimal. It also has register PC, which is a program counter, which is

responsible for keeping the instructions that will be executed next.

The screenshot displays an ARM assembly simulator interface. On the left, a list of assembly instructions is shown with line numbers 1 through 13. Line 8, the instruction `B again`, is highlighted in yellow, and a blue box labeled "Branch" points to it. The instructions are as follows:

```
1 MOV R0, #5 ; R0 is current number
2 MOV R1, #0 ; R1 is count of number of iterations
3 again ADD R1, R1, #1 ; increment number of iterations
4 ANDS R0, R0, #1 ; test whether R0 is odd
5 BEQ even
6 ADD R0, R0, R0, LSL #1 ; if odd, set R0 = R0 * 2
7 ADD R0, R0, #1 ; and repeat (guaranteed R0 > 1)
8 B again
9 even MOV R0, R0, ASR #1 ; if even, set R0 = R0 / 2
10 SUBS R7, R0, #1 ; and repeat if R0 != 1
11 BNE again
12 halt B halt ; infinite loop to stop computation
13
```

On the right side, a table shows the current values of registers R0 through R13, the Link Register (LR), and the Program Counter (PC). Each row includes the register name, its decimal value, and buttons to view the value in binary or hexadecimal.

Register	Value	Dec	Bin	Hex
R0	0x4			
R1	0x1			
R2	0x0			
R3	0x0			
R4	0x0			
R5	0x0			
R6	0x0			
R7	0x0			
R8	0x0			
R9	0x0			
R10	0x0			
R11	0x0			
R12	0x0			
R13	0xFF000000			
LR	0x0			
PC	0x24			

At the bottom of the simulator, a status bar shows "Clock Cycles" and "Current Instruction: 3 Total: 10". Below this, a row of status bits (CPSR Status Bits (NZCV)) is visible, showing values for flags C, Z, N, and V.

Now from lines 3 to line 11 is the while loop and the if else condition. First it increments the numbers of iterations. Then it tests if the value of register R0 is even or odd. If it is odd, then lines 6 and 7 are executed else lines 9 and 10 are executed. The changes in all the register can also be seen on the right side.

```

1      MOV     R0, #5      ; R0 is current number
2      MOV     R1, #0      ; R1 is count of number of iterations
3 again ADD     R1, R1, #1  ; increment number of iterations
4      ANDS    R0, R0, #1   ; test whether R0 is odd
5      BEQ     even
6      ADD     R0, R0, LSL #1 ; if odd, set R0 = R0 + (R0 << 1)
7      ADD     R0, R0, #1   ; and repeat (guaranteed R0 > 1)
8      B       again
9 even MOV     R0, R0, ASR #1 ; if even, set R0 = R0 / 2
10     SUBS    R7, R0, #1   ; and repeat if R0 != 1
11     BNE     again
12 halt B       halt      ; infinite loop to stop computation
13

```

Register	Value	Dec	Bin	Hex
R0	0x0			Hex
R1	0x21			Hex
R2	0x0			Hex
R3	0x0			Hex
R4	0x0			Hex
R5	0x0			Hex
R6	0x0			Hex
R7	0xFFFFFFFF			Hex
R8	0x0			Hex
R9	0x0			Hex
R10	0x0			Hex
R11	0x0			Hex
R12	0x0			Hex
R13	0xFF000000			Hex
LR	0x0			Hex
PC	0x28			Hex

⌚ Clock Cycles
Current Instruction: 1 Total: 326

CSPR Status Bits (NZCV)

This piece of code will keep running because we do not have the terminate condition. It will keep running either the value is odd or even.

Example 3: Condition Codes

```

a = 40;
b = 25;
while (a != b) {
    if (a > b) a -= b;
    else      b -= a;
}

```

This is the code in C. It means that the operation should take place only when certain combinations of the combinations of the flags hold. You can specify the condition code by including it as part of the opcode.

ARM ISA:

Example 3: Condition Codes

The screenshot displays an ARM emulator interface. On the left, a list of assembly instructions is shown with line numbers 1 through 8. The instructions are: 1. MOV R0, #40 ; R0 is a, 2. MOV R1, #25 ; R1 is b, 3. again, 4. CMP R0, R1, 5. SUBGT R0, R0, R1, 6. SUBLT R1, R1, R0, 7. BNE again, 8. halt B halt. The instruction at line 2 is highlighted. On the right, a table lists the registers R0 through R13, LR, and PC, along with their current values in hexadecimal. R0 is 0x28, R1 is 0x19, and all other registers are 0x0. The PC is 0xC. At the bottom, a progress bar shows 'Clock Cycles' and 'Current Instruction: 1 Total: 2'.

Register	Value	Dec	Bin	Hex
R0	0x28			
R1	0x19			
R2	0x0			
R3	0x0			
R4	0x0			
R5	0x0			
R6	0x0			
R7	0x0			
R8	0x0			
R9	0x0			
R10	0x0			
R11	0x0			
R12	0x0			
R13	0xFF000000			
LR	0x0			
PC	0xC			

This is the ARM ISA. On the left side of the screen are the ARM instructions for Condition Codes. The right side has all the register that are being used in this problem. This is the beginning of the code. So, you can see the registers R0 and R1 are initialized with their values. The values are in hexadecimal. It also has register PC, which is a program counter, which is responsible for keeping the instructions that will be executed next.

Reset to continue editing code

1MOV R0, #40 ; R0 is a

2MOV R1, #25 ; R1 is b

3againCMP R0, R1

4SUBGT R0, R0, R1

5SUBLT R1, R1, R0

6BNE again

7haltB halt

8

R0	0xF	Dec	Bin	Hex
R1	0x19	Dec	Bin	Hex
R2	0x0	Dec	Bin	Hex
R3	0x0	Dec	Bin	Hex
R4	0x0	Dec	Bin	Hex
R5	0x0	Dec	Bin	Hex
R6	0x0	Dec	Bin	Hex
R7	0x0	Dec	Bin	Hex
R8	0x0	Dec	Bin	Hex
R9	0x0	Dec	Bin	Hex
R10	0x0	Dec	Bin	Hex
R11	0x0	Dec	Bin	Hex
R12	0x0	Dec	Bin	Hex
R13	0xFF000000	Dec	Bin	Hex
LR	0x0	Dec	Bin	Hex
PC	0x14	Dec	Bin	Hex

Clock Cycles

Current Instruction: 1 Total: 4

CSPR Status Bits (NZCV)

0010

Line 3 of the code compares the value of the registers R0 and R1, stores it in register R0. Then line 4 and 5 checks which values is greater and lesser, then it subtracts the values from the registers.

Reset to continue editing code

1MOV R0, #40 ; R0 is a

2MOV R1, #25 ; R1 is b

3againCMP R0, R1

4SUBGT R0, R0, R1

5SUBLT R1, R1, R0

6BNE again

7haltB halt

8

Branch

R0	0x5	Dec	Bin	Hex
R1	0x5	Dec	Bin	Hex
R2	0x0	Dec	Bin	Hex
R3	0x0	Dec	Bin	Hex
R4	0x0	Dec	Bin	Hex
R5	0x0	Dec	Bin	Hex
R6	0x0	Dec	Bin	Hex
R7	0x0	Dec	Bin	Hex
R8	0x0	Dec	Bin	Hex
R9	0x0	Dec	Bin	Hex
R10	0x0	Dec	Bin	Hex
R11	0x0	Dec	Bin	Hex
R12	0x0	Dec	Bin	Hex
R13	0xFF000000	Dec	Bin	Hex
LR	0x0	Dec	Bin	Hex
PC	0x1C	Dec	Bin	Hex

Clock Cycles

Current Instruction: 3 Total: 26

CSPR Status Bits (NZCV)

1000

At the end, the value of register R0, R1 have the same value of 5.