

COS10004 – Computer System

Name: Phan Vũ – Student Id: 104222099

LAB 8

8.1.1

```
MOV R0, #15
STR R0, .WriteUnsignedNum
MOV R0, #msg1
STR R0, .WriteString
HALT
```

msg1: .ASCIZ "remaining\n"

The screenshot displays a computer system simulator interface with a blue border. The main window is divided into three sections: Program, Processor, and Input/Output.

Program Section: A list of assembly instructions is shown, with line 5 highlighted in orange. The instructions are:

```
1 MOV R0, #15
2 STR R0, .WriteUnsignedNum
3 MOV R0, #msg1
4 STR R0, .WriteString
5 HALT
6 msg1: .ASCIZ "remaining\n"
```

Processor Section: A table of processor registers and status bits is displayed. The PC (Program Counter) is 20, LR (Link Register) is 0, and SP (Stack Pointer) is 1048576. The R12 register is 0, and R11 through R0 are all 0. The Count register is 5. The Current Instruction is empty. The Status bits are NZCV, with values 0000.

Input/Output Section: The output shows "15 remaining" and "Program HALTED. STOP, LOAD or EDIT".

At the bottom of the interface, there are three buttons: Load, Save, and Edit.

8.1.2.

```
MOV R0, #15
STR R0, .WriteUnsignedNum
```

```

MOV R0, #msg1
STR R0, .WriteString
MOV R0, #msg2
STR R0, .WriteString
LDR R1, .InputNum
HALT

```

msg1: .ASCIZ "remaining\n"

msg2: .ASCIZ "How many do you want to remove (1-3)?"

The screenshot shows a debugger window with three main panels:

- Program:** A list of assembly instructions. Line 8, `HALT`, is highlighted in orange.


```

1  MOV R0, #15
2  STR R0, .WriteUnsignedNum
3  MOV R0, #msg1
4  STR R0, .WriteString
5  MOV R0, #msg2
6  STR R0, .WriteString
7  LDR R1, .InputNum
8  HALT
9  msg1: .ASCIZ "remaining\n"
10 msg2: .ASCIZ "How many do you want to remove (1-3)?"
      
```
- Processor:** A table of registers and control elements.

PC	32
LR	0
SP	1048576
R12	0
R11	0
R10	0
R9	0
R8	0
R7	0
R6	0
R5	0
R4	0
R3	0
R2	0
R1	2
R0	43

 Below the registers are controls for **Count** (set to 8), **Current Instruction**, and **Status bits** (NZCV: 0000).
- Input/Output:** A text area showing program output:


```

15 remaining
How many do you want to remove (1-3)?
Program HALTED. STOP, LOAD or EDIT
      
```

 Below this is a text input field containing the number `2`.

At the bottom of the window are buttons for **Load**, **Save**, and **Edit**.

8.1.3.

```

MOV R0, #15
STR R0, .WriteUnsignedNum
MOV R1, #msg1
STR R1, .WriteString
MOV R1, #msg2
STR R1, .WriteString
LDR R2, .InputNum

```

```

SUB R0, R0, R2
STR R0, .WriteUnsignedNum
MOV R1, #msg1
STR R1, .WriteString
HALT

```

msg1: .ASCIZ "remaining\n"

msg2: .ASCIZ "How many do you want to remove (1-3)?\n"

The screenshot displays an ARM assembly simulator interface. The main window is titled "Program" and shows a list of assembly instructions. The instruction at line 12, "HALT", is highlighted in orange. Below the program window, there are three buttons: "Load", "Save", and "Edit".

To the right of the program window is the "Processor" panel. It displays the following registers and values:

PC	48
LR	0
SP	1048576
R12	0
R11	0
R10	0
R9	0
R8	0
R7	0
R6	0
R5	0
R4	0
R3	0
R2	3
R1	48
R0	12

Below the registers, there are control buttons: a play button, a pause button, a stop button, a step forward button, a step back button, and a settings gear icon. There are also fields for "Count" (value 12), "Current Instruction" (empty), and "Status bits" (value 0000).

At the bottom right is the "Input/Output" panel. It shows the prompt "How many do you want to remove (1-3)?" and the input "12 remaining". Below this, it says "Program HALTED. STOP, LOAD or EDIT". At the very bottom, there is a small input field containing the number "3".

8.2.1

```

MOV R0, #15
STR R0, .WriteUnsignedNum
MOV R1, #msg1
STR R1, .WriteString

```

Loop:

```

MOV R1, #msg2
STR R1, .WriteString
LDR R2, .InputNum

```

```

SUB R0, R0, R2
STR R0, .WriteUnsignedNum
MOV R1, #msg1
STR R1, .WriteString
B Loop
HALT

```

msg1: .ASCIZ "remaining\n"

msg2: .ASCIZ "How many do you want to remove (1-3)?\n"

Program

```

1  MOV R0, #15
2  STR R0, .WriteUnsignedNum
3  MOV R1, #msg1
4  STR R1, .WriteString
5  Loop:
6  MOV R1, #msg2
7  STR R1, .WriteString
8  LDR R2, .InputNum
9  SUB R0, R0, R2
10 STR R0, .WriteUnsignedNum
11 MOV R1, #msg1
12 STR R1, .WriteString
13 B Loop
14 HALT
15 msg1: .ASCIZ "remaining\n"
16 msg2: .ASCIZ "How many do you want to remove (1-3)?\n"

```

Processor

PC	28
LR	0
SP	1048576
R12	0
R11	0
R10	0
R9	0
R8	0
R7	0
R6	0
R5	0
R4	0
R3	0
R2	1
R1	63
R0	14

Count

Current Instruction

Status bits NZCV

Input/Output

```

How many do you want to remove (1-3)?
14 remaining
How many do you want to remove (1-3)?

```

Input expected

Program

```

1      MOV R0, #15
2      STR R0, .WriteUnsignedNum
3      MOV R1, #msg1
4      STR R1, .WriteString
5 Loop:
6      MOV R1, #msg2
7      STR R1, .WriteString
8      LDR R2, .InputNum
9      SUB R0, R0, R2
10     STR R0, .WriteUnsignedNum
11     MOV R1, #msg1
12     STR R1, .WriteString
13     B Loop
14     HALT
15 msg1: .ASCIIZ "remaining\n"
16 msg2: .ASCIIZ "How many do you want to remove (1-3)?\n"

```

Processor

PC	0x0000001c
LR	0x00000000
SP	0x00100000
R12	0x00000000
R11	0x00000000
R10	0x00000000
R9	0x00000000
R8	0x00000000
R7	0x00000000
R6	0x00000000
R5	0x00000000
R4	0x00000000
R3	0x00000000
R2	0x00000003
R1	0x0000003f
R0	0x0000000b

Count 23

Current Instruction

Status bits N Z C V
0 0 0 0

Input/Output

How many do you want to remove (1-3)?
 11 remaining
 How many do you want to remove (1-3)?

Input expected

If you enter a number that takes the number of matchsticks remaining beyond 0 (i.e., into negative values), the remaining number will still be calculated as normal, for example we are having 10 matchsticks and we remove -1 matchsticks, then the new number will be 11 matchsticks.

8.2.2

(a) $0 < R2 < 4$

(b)

Two assembly instructions could be used to create a branch that only occurs under this condition: BGT and BLT.

BGT: Z clear, N and V the same

BLT: N and V differ

(c) If the first condition is not met ($R2 > 0$), and R2 is negative then N =

1. If the first condition is not met ($R2 > 0$), and $R2 = 0$ then $Z = 1$.

If the second condition is not met ($R2 < 4$), and $R2 > 4$ then $C = 1$. If the second condition is not met ($R2 < 4$), and $R2 = 4$ then both $Z = 1$ and $C = 1$.

(d)

```
MOV R0, #15
STR R0, .WriteUnsignedNum
MOV R1, #msg1
STR R1, .WriteString
Loop:
    MOV R1, #msg2
    STR R1, .WriteString
    LDR R2, .InputNum
start:
    CMP R2, #0
    BGT else1      // if R2 > 0 then jump to label else1
    B invalid1
else1:
    CMP R2, #4
    BGT invalid1   // if R2 > 3 then jump to label invalid1
    BLT cont       // if R2 < 4 then jump to label cont
invalid1:
    MOV R1, #msg3
    STR R1, .WriteString
    B start
cont:
    SUB R0, R0, R2
    STR R0, .WriteUnsignedNum
    MOV R1, #msg1
    STR R1, .WriteString
    B Loop
    HALT
msg1: .ASCIZ "remaining\n"
msg2: .ASCIZ "How many do you want to remove (1-3)?\n"
msg3: .ASCIZ "Please input a valid number!\n"
```

Program

```

1      MOV R0, #15
2      STR R0, .WriteUnsignedNum
3      MOV R1, #msg1
4      STR R1, .WriteString
5 Loop:
6      MOV R1, #msg2
7      STR R1, .WriteString
8      LDR R2, .InputNum
9 start:
10     CMP R2, #0
11     BGT else1           // if R2 > 0 then jump to label else1
12     B invalid1
13 else:
14     CMP R2, #4
15     BGT invalid1       // if R2 > 3 then jump to label invalid1
16     BLT cont           // if R2 < 4 then jump to label cont
17 invalid1:
18     MOV R1, #msg3
19     STR R1, .WriteString
20     B start
21 cont:
22     SUB R0, R0, R2
23     STR R0, .WriteUnsignedNum
24     MOV R1, #msg1
25     STR R1, .WriteString
26     B Loop
27     HALT
28 msg1: .ASCIZ "remaining\n"
29 msg2: .ASCIZ "How many do you want to remove (1-3)?\n"
30 msg3: .ASCIZ "Please input a valid number!\n"

```

Processor

PC	0x0000001c
LR	0x00000000
SP	0x00100000
R12	0x00000000
R11	0x00000000
R10	0x00000000
R9	0x00000000
R8	0x00000000
R7	0x00000000
R6	0x00000000
R5	0x00000000
R4	0x00000000
R3	0x00000000
R2	0x00000003
R1	0x00000063
R0	0x0000000c

Count 20

Current Instruction

Status bits NZCV 1000

Input/Output

How many do you want to remove (1-3)?
 12 remaining
 How many do you want to remove (1-3)?

Input expected

8.3.1

(a)

LSL R4, R4, #30

LSR R4, R4, #30

(b)

select:

LDR R4, .Random

LSL R4, R4, #30

LSR R4, R4, #30

CMP R4, #0

BGT conti

B select

conti:

STR R4, .WriteUnsignedNum

HALT

Program

```

1 select:
2     LDR R4, .Random
3     LSL R4, R4, #30
4     LSR R4, R4, #30
5     CMP R4, #0
6     BGT conti
7     B select
8 conti:
9     STR R4, .WriteUnsignedNum
10    HALT

```

Load

Save

Edit

Processor

PC	0x00000020
LR	0x00000000
SP	0x00100000
R12	0x00000000
R11	0x00000000
R10	0x00000000
R9	0x00000000
R8	0x00000000
R7	0x00000000
R6	0x00000000
R5	0x00000000
R4	0x00000002
R3	0x00000000
R2	0x00000000
R1	0x00000000
R0	0x00000000

▶ || ◻

◀ ▶ ⚙

Count

7

Current Instruction

Status bits

NZCV
0000

Input/Output

2

Program HALTED. STOP, LOAD or EDIT

8.3.2.

MOV R0, #3

select:

LDR R4, .Random

LSL R4, R4, #30

LSR R4, R4, #30

CMP R4, #0

BGT conti

B select

conti:

CMP R4, R0

BGT select

B continue

continue:

STR R4, .WriteUnsignedNum

HALT

The screenshot shows a debugger window with a blue header. The main area is divided into three sections: Program, Processor, and Input/Output.

Program: The assembly code is displayed with line numbers 1 through 15. Line 15 is highlighted in orange and contains the instruction `HALT`. The code is as follows:

```
1 MOV R0, #3
2 select:
3   LDR R4, .Random
4   LSL R4, R4, #30
5   LSR R4, R4, #30
6   CMP R4, #0
7   BGT conti
8   B select
9 conti:
10  CMP R4, R0
11  BGT select
12  B continue
13 continue:
14  STR R4, .WriteUnsignedNum
15  HALT
```

Processor: The register values are listed on the left, and the status bits are on the right. The registers are:

PC	0x00000030
LR	0x00000000
SP	0x00100000
R12	0x00000000
R11	0x00000000
R10	0x00000000
R9	0x00000000
R8	0x00000000
R7	0x00000000
R6	0x00000000
R5	0x00000000
R4	0x00000003
R3	0x00000000
R2	0x00000000
R1	0x00000000
R0	0x00000003

The status bits are: `NZCV 0110`.

Input/Output: The output window shows the text: `3` and `Program HALTED. STOP, LOAD or EDIT`.

At the bottom of the Program section, there are three buttons: `Load`, `Save`, and `Edit`.

8.4.1.

```
MOV R0, #15
STR R0, .WriteUnsignedNum
MOV R1, #msg1
STR R1, .WriteString
```

Loop:

```
MOV R1, #msg2
STR R1, .WriteString
LDR R2, .InputNum
```

start:

```
CMP R2, #0
BGT else1      // if R2 > 0 then jump to label else1
B invalid1
```

else1:

```

    CMP R2, #4
    BGT invalid1    // if R2 > 3 then jump to label invalid1
    BLT cont        // if R2 < 4 then jump to label cont
invalid1:
    MOV R1, #msg3
    STR R1, .WriteString
    B start
cont1:
    CMP R0, R2
    BLT invalid1
    B cont2
cont2:
    SUB R0, R0, R2
    STR R0, .WriteUnsignedNum
    MOV R1, #msg1
    STR R1, .WriteString
    MOV R1, #msg5
    STR R1, .WriteString
    B select
select:
    LDR R4, .Random
    LSL R4, R4, #30
    LSR R4, R4, #30
    CMP R4, #0
    BGT conti
    B select
conti:
    CMP R4, R0
    BGT select
    B continue
continue:
    SUB R0, R0, R4
    STR R0, .WriteUnsignedNum
    MOV R1, #msg1
    STR R1, .WriteString

```

```

    MOV R1, #msg4
    STR R1, .WriteString
    B Loop
    HALT
msg1: .ASCIZ "remaining\n"
msg2: .ASCIZ "How many do you want to remove (1-3)?\n"
msg3: .ASCIZ "Please input a valid number!\n"
msg4: .ASCIZ "It's your turn!\n"
msg5: .ASCIZ "It's computer's turn!\n"

select:
    LDR R4, .Random
    LSL R4, R4, #30
    LSR R4, R4, #30
    CMP R4, #0
    BGT conti
    B select
conti:
    CMP R4, R0
    BGT select
    B continue
//

    SUB R0, R0, R4
    STR R0, .WriteUnsignedNum
    MOV R1, #msg1
    STR R1, .WriteString

```