

Task4:

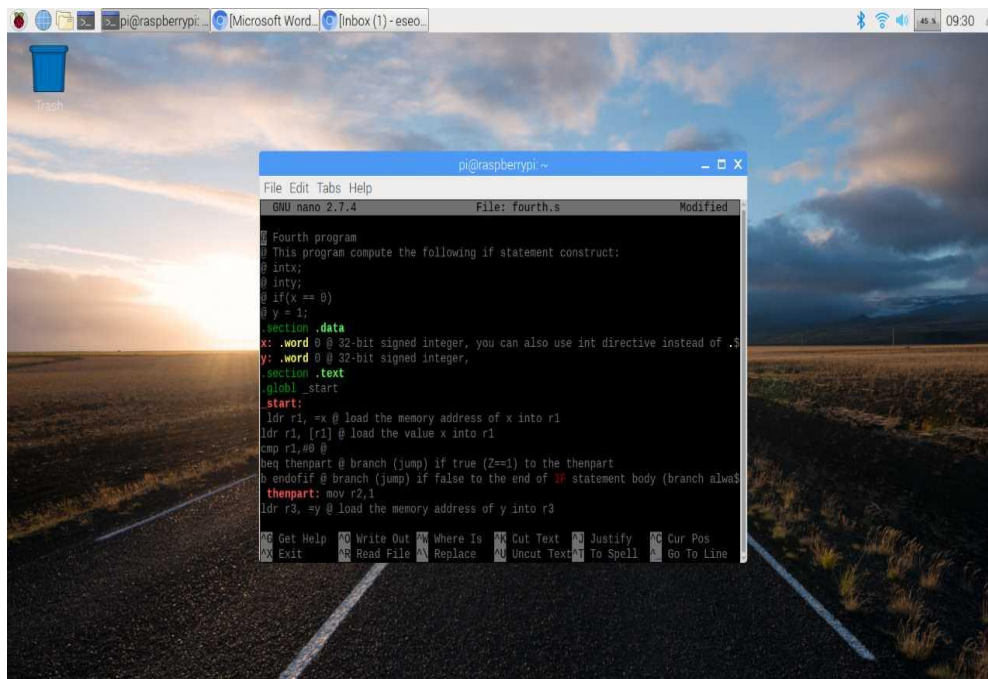
ARM assembler

These instructions assume that you have a Raspberry Pi up and running. After starting up your Raspberry Pi, you should have a GUI interface to Raspbian, the Linux variant operating system for these little machines. For this tutorial, you will primarily use the terminal window and type commands into it. You will also use an editor (more on that below).

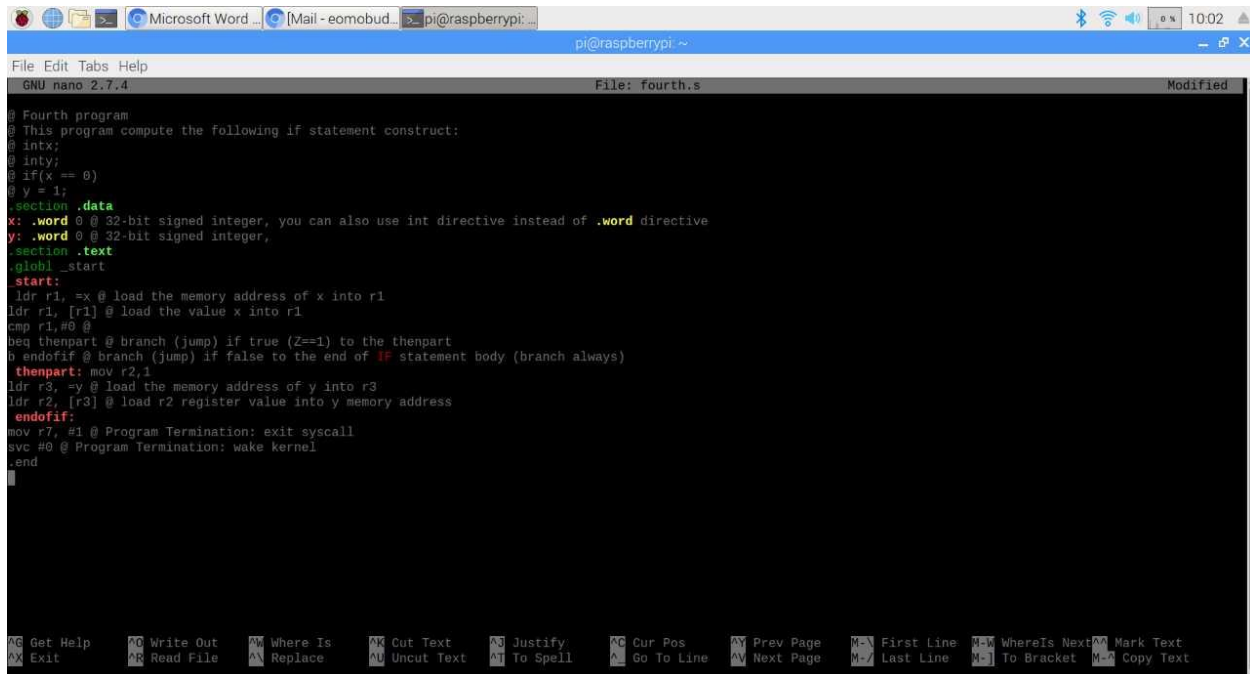
a) Part1: fourth program:

1. Open the **Terminal** (command line) in Raspberry Pi, click on the 4th icon to the left on the top bar.
2. Type the following in the terminal window, along with the file name:
nano fourth.s

Note: Control-w writes the file (save); control-x exits the editor.



- We had to search how to use scrot to save and screenshot our work.
 - This screenshot shows what happened after we used scrot.
3. Write the following program using nano and save it (control-w)
See appendix section to see how signed integer is represented
 - The screenshot below shows the program inputted into the Raspberry Pi.



```
@ Fourth program
@ This program compute the following if statement construct:
@ intx;
@ inty;
@ if(x == 0)
@ y = 1;

.section .data
x: .word 0 @ 32-bit signed integer, you can also use int directive instead of .word directive
y: .word 0 @ 32-bit signed integer,
.section .text
.globl _start
_start:
    ldr r1, =x @ load the memory address of x into r1
    ldr r1, [r1] @ load the value x into r1
    cmp r1, #0 @
    bge thenpart @ branch (jump) if true (Z=1) to the thenpart
    bne endif @ branch (jump) if false to the end of if statement body (branch always)
    thenpart: mov r2, 1
    ldr r3, =y @ load the memory address of y into r3
    ldr r2, [r3] @ load r2 register value into y memory address
    endif:
    mov r7, #1 @ Program Termination: exit syscall
    svc #0 @ Program Termination: wake kernel
    .end
```

- According to the assignment, we had to move all the values between the registers.
- This was the program after we typed it into the Raspberry Pi.
- We had to make some slight changes in order to get the program to run as there were some typos in the instructions.

4. To **assemble** the file, type the following command:

Debug your program by doing the following in the terminal window:

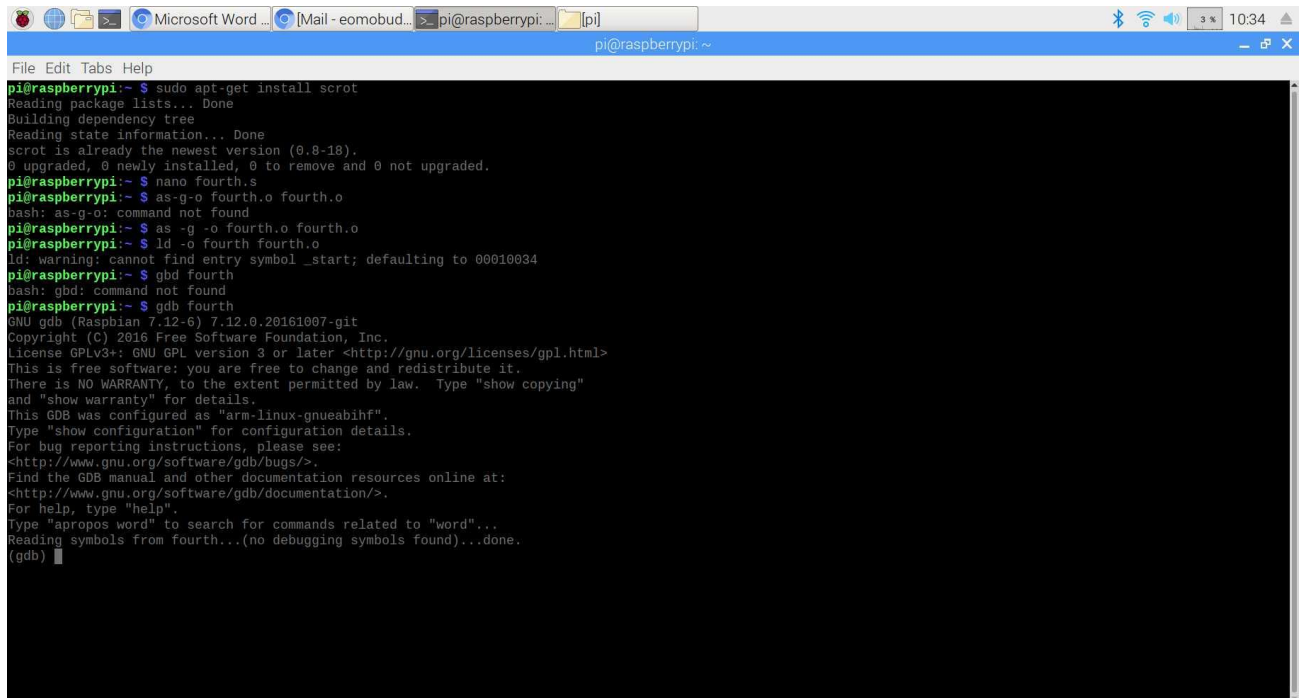
as -g -o fourth.o fourth.s

The linker command stays the same: **ld -o fourth fourth.o**

To launch the GNU Debugger, type the command **gdb** followed by the executable file name at the prompt:

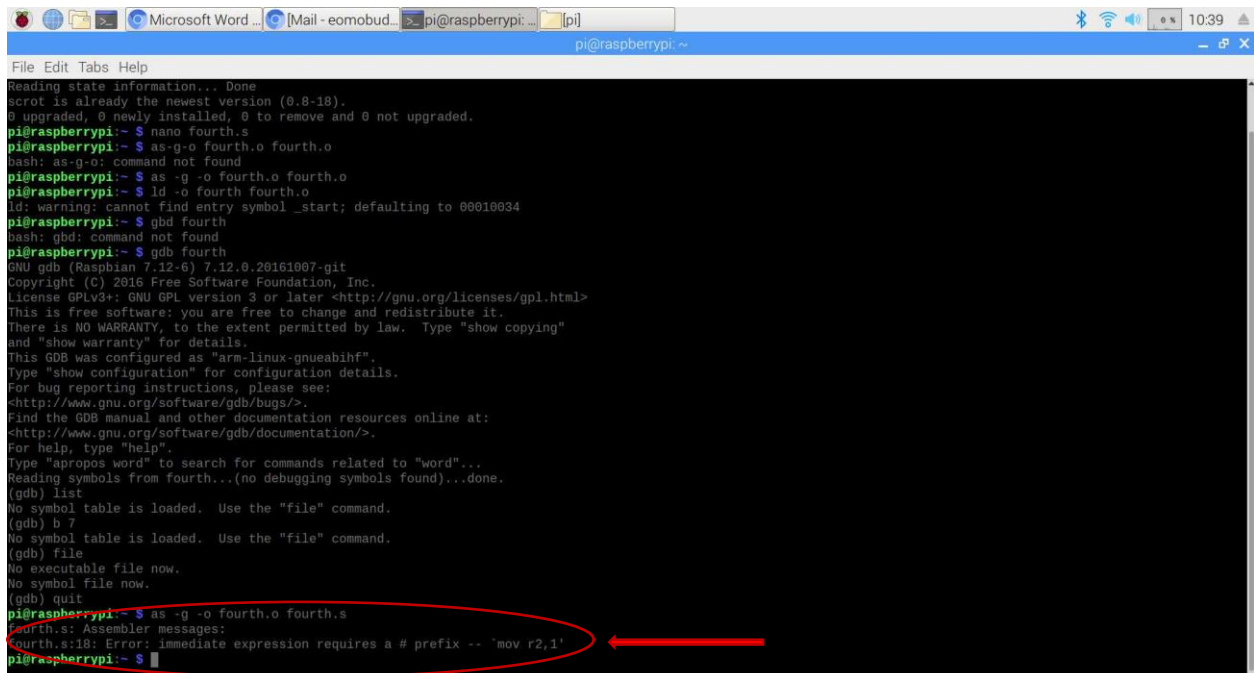
gdb fourth

After displaying the license and warranty statements, the prompt is shown as (gdb). See the following figure:



```
pi@raspberrypi:~$ sudo apt-get install scrot
Reading package lists... Done
Building dependency tree
Reading state information... Done
scrot is already the newest version (0.8-18).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@raspberrypi:~$ nano fourth.s
pi@raspberrypi:~$ as -g -o fourth.o fourth.s
bash: as-g-o: command not found
pi@raspberrypi:~$ as -g -o fourth.o fourth.o
pi@raspberrypi:~$ ld -o fourth fourth.o
ld: warning: cannot find entry symbol _start; defaulting to 00010034
pi@raspberrypi:~$ gdb fourth
bash: gdb: command not found
pi@raspberrypi:~$ gdb fourth
GNU gdb (Raspbian 7.12-6) 7.12.0.20161007-git
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "arm-linux-gnueabi".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from fourth...(no debugging symbols found)...done.
(gdb)
```

- We ran into a few problems trying to assemble and link the program. These screenshots show the errors we encountered.
- We had to fix the program so that it could find “_start”



```
pi@raspberrypi:~$ nano fourth.s
pi@raspberrypi:~$ as -g -o fourth.o fourth.s
bash: as-g-o: command not found
pi@raspberrypi:~$ as -g -o fourth.o fourth.o
pi@raspberrypi:~$ ld -o fourth fourth.o
ld: warning: cannot find entry symbol _start; defaulting to 00010034
pi@raspberrypi:~$ gdb fourth
bash: gdb: command not found
pi@raspberrypi:~$ gdb fourth
GNU gdb (Raspbian 7.12-6) 7.12.0.20161007-git
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "arm-linux-gnueabi".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from fourth...(no debugging symbols found)...done.
(gdb) list
No symbol table is loaded. Use the "file" command.
(gdb) b 7
No symbol table is loaded. Use the "file" command.
(gdb) file
No executable file now.
No symbol file now.
(gdb) quit
pi@raspberrypi:~$ as -g -o fourth.o fourth.s
fourth.s:18: Error: immediate expression requires a # prefix -- 'mov r2,1'
pi@raspberrypi:~$
```

- After an hour of trying to fix the problems, we finally managed to fix one, however, another one popped up.

- There was an error with the part of the code “mov r2,1” so we had to fix that as well.
- We fixed it by rewording the code and adding a Direct Operand
- After that process was complete we used (gdb) stepi and this is shown in the screenshot below.

```

File Edit Tabs Help
3  @ intx;
4  @ inty;
5  @ if(x == 0)
6  @ y = 1;
7  .section .data
8  x: word 0 @ 32-bit signed integer, you can also use int directive instead of word directive
9  y: word 0 @ 32-bit signed integer,
10 .section .text
(gdb) b 7
Breakpoint 1 at 0x10078: File fourth.s, line 7.
(gdb) run
Starting program: /home/pi/fourth
Breakpoint 1, _start () at fourth.s:14
14  ldr r1, [r1] @ load the value x into r1
(gdb) stepi
15  cmp r1,#0 @
(gdb) x/iw 0x10078
0x10078 <_start+4>: 0xe5911000
(gdb) info registers
r0      0x0      0
r1      0x0      0
r2      0x0      0
r3      0x0      0
r4      0x0      0
r5      0x0      0
r6      0x0      0
r7      0x0      0
r8      0x0      0
r9      0x0      0
r10     0x0      0
r11     0x0      0
r12     0x0      0
sp      0x7efff070 0x7efff070
lr      0x0      0
pc      0x1007c 0x1007c <_start+8>
cpsr    0x10     16
(gdb)

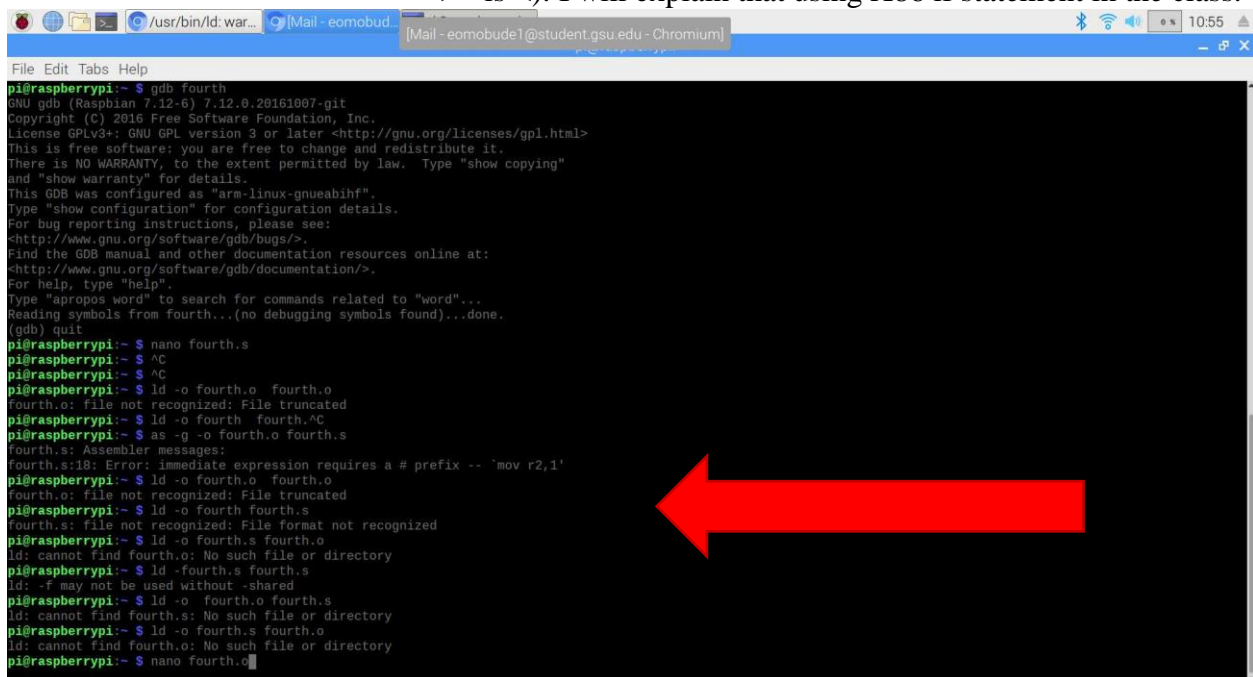
```

- Before we could go through the program, we had to set breakpoints.
- These were set in the program so that we are able to see what data is being moved to the registers and what values are getting changed as the program progresses.
- We checked the values by using ‘stepi’ command to see each line running one by one and also ‘info registers’ to check the content of the registers.
- Breakpoint 1 was created at line #7
- Breakpoint 2 was created at line #14

b) **Part2:** Using the fourth.s program as a reference to update, assemble, link, run, and debug the new program.

- Is the program in part one efficient?
 - No the program is not efficient because it includes an instruction that does not allow it to run correctly. To be specific, the code contains back-to-back branches (**beq** followed by **b**). We want to avoid this, since branches may cause a delay slot.
- What did we do?
 - We replaced **beq** with **bne**(branch on not equal ($Z==0$)) and we removed the line **b endofif**

Note: What you did here is you jumped when the condition was false which is different from what we do in the high level languages. In Other words, we reversed the Boolean expression using DE Morgan Law ($< \text{ is } \geq$, $> \text{ is } \leq$, $== \text{ is } !=$, $\leq \text{ is } >$ and $\geq \text{ is } <$). I will explain that using X86 if statement in the class.



```
pi@raspberrypi:~$ gdb fourth
GNU gdb (Raspbian 7.12-6) 7.12.0.20161007-git
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "arm-linux-gnueabi".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from fourth...(no debugging symbols found)...done.
(gdb) quit
pi@raspberrypi:~$ nano fourth.s
pi@raspberrypi:~$ ^C
pi@raspberrypi:~$ ^C
pi@raspberrypi:~$ ld -o fourth.o fourth.o
fourth.o: file not recognized: File truncated
pi@raspberrypi:~$ ld -o fourth fourth.o
pi@raspberrypi:~$ as -g -o fourth.o fourth.s
fourth.s: Assembler messages:
fourth.s:18: Error: immediate expression requires a # prefix -- 'mov r2,1'
pi@raspberrypi:~$ ld -o fourth.o fourth.o
fourth.o: file not recognized: File truncated
pi@raspberrypi:~$ ld -o fourth fourth.s
fourth.s: file not recognized: File format not recognized
pi@raspberrypi:~$ ld -o fourth.s fourth.o
ld: cannot find fourth.o: No such file or directory
pi@raspberrypi:~$ ld -fourth.s fourth.s
ld: -f may not be used without -shared
pi@raspberrypi:~$ ld -o fourth.o fourth.s
ld: cannot find fourth.s: No such file or directory
pi@raspberrypi:~$ ld -o fourth.s fourth.o
ld: cannot find fourth.o: No such file or directory
pi@raspberrypi:~$ nano fourth.o
```

-After we implemented this step of removing the “b endofif” line, we had to fix a few of the errors In the code because it would not run. We had to add an immediate prefix. We also changed “1” to “#1” in line s:18 and then tried to run the program again.

```
File Edit Tabs Help
3  @ intx;
4  @ inty;
5  @ if(x == 0)
6  @ y = 1;
7  .section .data
8  x: .word 0 @ 32-bit signed integer, you can also use int directive instead of .word directive
9  y: .word 0 @ 32-bit signed integer,
10 .section .text
(gdb) b 7
Breakpoint 1 at 0x10078: file fourth.s, line 7.
(gdb) run
Starting program: /home/pi/fourth
Breakpoint 1, _start () at fourth.s:14
14  ldr r1, [r1] @ load the value x into r1
(gdb) stepi
15  cmp r1, #0 @
(gdb) x/ixw 0x10078
0x10078 <_start+4>: 0xe5911000
(gdb) info registers
r0          0x0      0
r1          0x0      0
r2          0x0      0
r3          0x0      0
r4          0x0      0
r5          0x0      0
r6          0x0      0
r7          0x0      0
r8          0x0      0
r9          0x0      0
r10         0x0      0
r11         0x0      0
r12         0x0      0
sp          0x7efff070 0x7efff070
lr          0x0      0
pc          0x1007c 0x1007c <_start+8>
cpsr        0x10      16
(gdb) 
```

-The z flag is in the cpsr register which is at the bottom. The Z flag is 1 because the outcome in r1 is 0.

-There is a value of 16 in one of the registers.

c) **Part3:** Using the fourth.s program as a reference to edit, assemble, link, run, and debug the new program.

Write a program that calculates the following expression:

```
if X <= 3
X = X - 1
else
X = X-2
```

-We ran into a lot of problems trying to get the program to run. It took hours of coding and reprogramming to finally create a program that worked. This is what we came up with.

```
pi@raspberrypi:~$ nano fourth.s
GNU nano 2.7.4 File: fourth.s

@fourth
@ This program compute the following if statement construct:
@ if x<=3;
@ x=x-1;
@ else;
@ x=x+2;
@ X = 1;
.section .data
x: .word 1 @ 32-bit signed integer, you can also use int directive instead of .word directive

.section .text
.globl _start
_start:

    ldr r1, =x @ load the memory address of x into r1
    ldr r1, [r1] @ load the value x into r1
    cmp r1, #3 @
    ble thenpart @ branch (jump) if less than or equal x<=3 (( Z or N xor V)=1)
    sub r1, r1, #2 @sub 1 from r1 and then move it to r1
    b endif
thenpart: mov r2, #1
    sub r1, r2, #1 @ sub 1 from r2 and then move to r1
endif:
    mov r7, r1 @ Program Termination: exit syscall
    svc #3 @ Program Termination: wake kernel
    .end

[Read 30 lines]
Get Help Write Out Where Is Cut Text Justify Cur Pos Prev Page First Line WhereIs Next Mark Text
Exit Read File Replace Uncut Text To Spell Go To Line Next Page Last Line To Bracket Copy Text
```

-We used the Fourth.s program as a guide and then replaced some things. For example, we removed the second variable and just left the code with one variable (x). We ended up using “sub r1, r1, #2” and “sub r1, r2, #1” to get the final result stored into the r1 register.

```
pi@raspberrypi:~$ nano fourth.s
GNU nano 2.7.4 File: fourth.s

@fourth
@ This program compute the following if statement construct:
@ if x<=3;
@ x=x-1;
@ else;
@ x=x+2;
@ X = 1;
.section .data
x: .word 1 @ 32-bit signed integer, you can also use int directive instead of .word directive

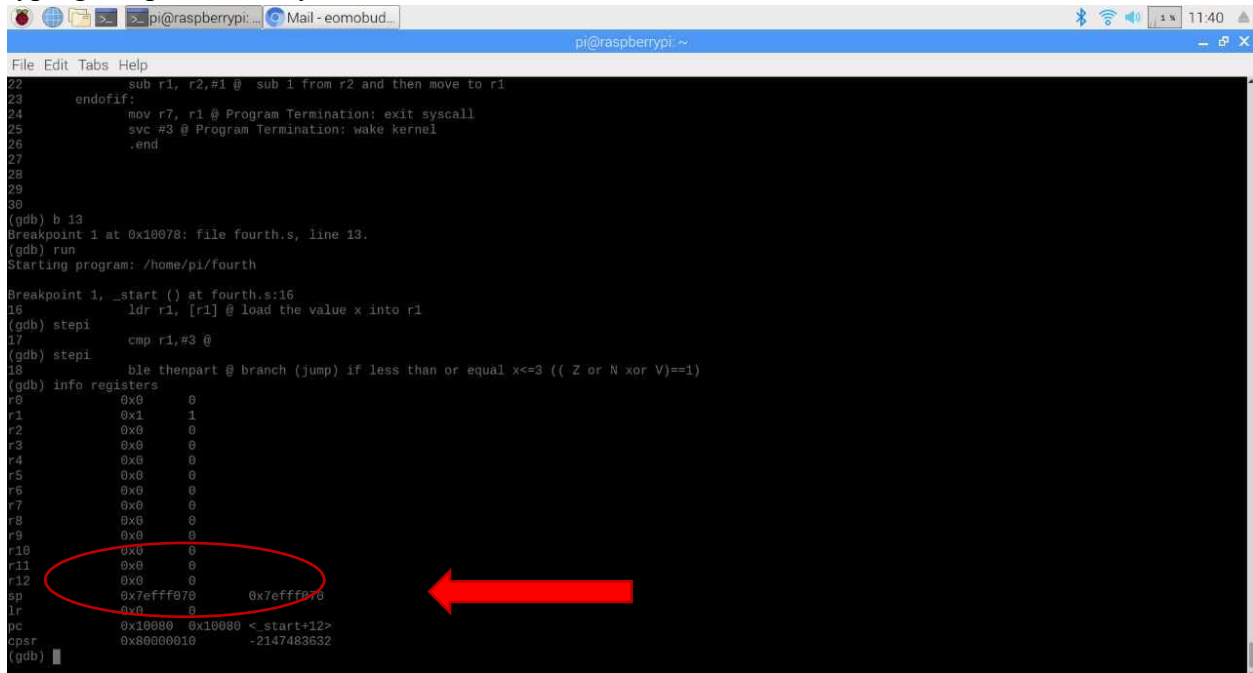
.section .text
.globl _start
_start:

    ldr r1, =x @ load the memory address of x into r1
    ldr r1, [r1] @ load the value x into r1
    cmp r1, #3 @
    ble thenpart @ branch (jump) if less than or equal x<=3 (( Z or N xor V)=1)
    sub r1, r1, #2 @sub 1 from r1 and then move it to r1
    b endif
thenpart: mov r2, #1
    sub r1, r2, #1 @ sub 1 from r2 and then move to r1
endif:
    mov r7, r1 @ Program Termination: exit syscall
    svc #3 @ Program Termination: wake kernel
    .end

pi@raspberrypi:~$ gcc -o fourth.o fourth.s
pi@raspberrypi:~$ ld -o fourth fourth.o
pi@raspberrypi:~$ gdb fourth
GNU gdb (Raspbian 7.12-6) 7.12.0.20161007-git
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "arm-linux-gnueabi".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from fourth...done.
(gdb) list
1  @fourth
2  @ This program compute the following if statement construct:
3  @ if x<=3;
4  @ x=x-1;
5  @ else;
6  @ x=x+2;
7  @ X = 1;
8  .section .data
9  x: .word 1 @ 32-bit signed integer, you can also use int directive instead of .word directive
10
(gdb) list
11 .section .text
12 .globl _start
13 _start:
```


-The program kept getting terminated due to that and it is shown at the top of this screenshot.

-We had to type “quit” several times and redo the whole process because we kept typing “stepi” too many times.



```
File Edit Tabs Help
22      sub r1, r2, #1 @ sub 1 from r2 and then move to r1
23  endif:
24      mov r7, r1 @ Program Termination: exit syscall
25      svc #3 @ Program Termination: wake kernel
26      .end
27
28
29
30
(gdb) b 13
Breakpoint 1 at 0x10078: file fourth.s, line 13.
(gdb) run
Starting program: /home/pi/fourth

Breakpoint 1, _start () at fourth.s:16
16      ldr r1, [r1] @ load the value x into r1
(gdb) stepi
17      cmp r1, #3 @
(gdb) stepi
18      ble thenpart @ branch (jump) if less than or equal x<=3 (( Z or N xor V)==1)
(gdb) info registers
r0          0x0      0
r1          0x1      1
r2          0x0      0
r3          0x0      0
r4          0x0      0
r5          0x0      0
r6          0x0      0
r7          0x0      0
r8          0x0      0
r9          0x0      0
r10         0x0      0
r11         0x0      0
r12         0x0      0
sp          0x7efff070 0x7efff070
lr          0x0      0
pc          0x10080 0x10080 <start+12>
cpsr       0x80000010 -2147483632
(gdb)
```

-We observed on the first command that it shows the stored value inside the memory(0x7efff070) but another command gives a long output with the same \x7efff\.

-Breakpoints were set at s:13 and s:16


```
pi@raspberrypi:~$ gdb
File Edit Tabs Help
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from fourth...done.
(gdb) list
1      @fourth
2      @ This program compute the following if statement construct:
3      @ if x<=3;
4      @ x=x-1;
5      @ else;
6      @ x=x-2;
7      @ X = 1;
8      .section .data
9      x: .word 1 @ 32-bit signed integer, you can also use int directive instead of .word directive
10
(gdb) list
11      .section .text
12      .globl _start
13      _start:
14
15          ldr r1, =x @ load the memory address of x into r1
16          ldr r1, [r1] @ load the value x into r1
17          cmp r1, #3 @
18          ble thenpart @ branch (jump) if less than or equal x<=3 (( Z or N xor V)=1)
19          sub r1, r1, #2 @sub 1 from r1 and then move it to r1
20          b endifif
(gdb) list
21      thenpart: mov r2, #1
22                sub r1, r2, #1 @ sub 1 from r2 and then move to r1
23      endifif:
24                mov r7, r1 @ Program Termination: exit syscall
25                svc #3 @ Program Termination: wake kernel
26                .end
27
28
29
30
(gdb) b 13
Breakpoint 1 at 0x10078: file fourth.s, line 13.
```

- We ran into a lot of problems after changing the variables for the new code.
- After a lot of trial and error, we were finally able to figure out how to get the code to assemble and run.
- We set a breakpoint at line s:13 :0x10078

Assume that X is 32-bit integer memory variables and assign X 1.

- **Use the debugger to verify the result in the memories and the Register.**
- **Report the X value in hex (as shown in the debugger) and Z flag as shown.**

```
pi@raspberrypi:~$ gdb
File Edit Tabs Help
22      sub r1, r2, #1 @ sub 1 from r2 and then move to r1
23      endifif:
24          mov r7, r1 @ Program Termination: exit syscall
25          svc #3 @ Program Termination: wake kernel
26          .end
27
28
29
30
(gdb) b 13
Breakpoint 1 at 0x10078: file fourth.s, line 13.
(gdb) run
Starting program: /home/pi/fourth
Breakpoint 1, _start () at fourth.s:16
16      ldr r1, [r1] @ load the value x into r1
(gdb) stepi
17      cmp r1, #3 @
(gdb) stepi
18      ble thenpart @ branch (jump) if less than or equal x<=3 (( Z or N xor V)=1)
(gdb) info registers
r0          0x0      0
r1          0x1      1
r2          0x0      0
r3          0x0      0
r4          0x0      0
r5          0x0      0
r6          0x0      0
r7          0x0      0
r8          0x0      0
r9          0x0      0
r10         0x0      0
r11         0x0      0
r12         0x0      0
sp          0x7efff070 0x7efff070
lr          0x0      0
pc          0x10078 <start+12>
cpsr       0x80000010 -2147483632
(gdb)
```

- We confirmed that r1 had the value of 1 as shown in this screenshot.
- The flags are shown in the bottom next to the cpsr line. This is where the values are stored in the registers.
- cpsr is the register where the total value(solution) of the equation is stored which is
- R1 contains the signed value 0
- R2 contains the first half of the program and the value is 0
- R3 contains the second half of the program and the value is 0

The most confusing part of this assignment was figuring out how to run the program and create my own program in part 3.