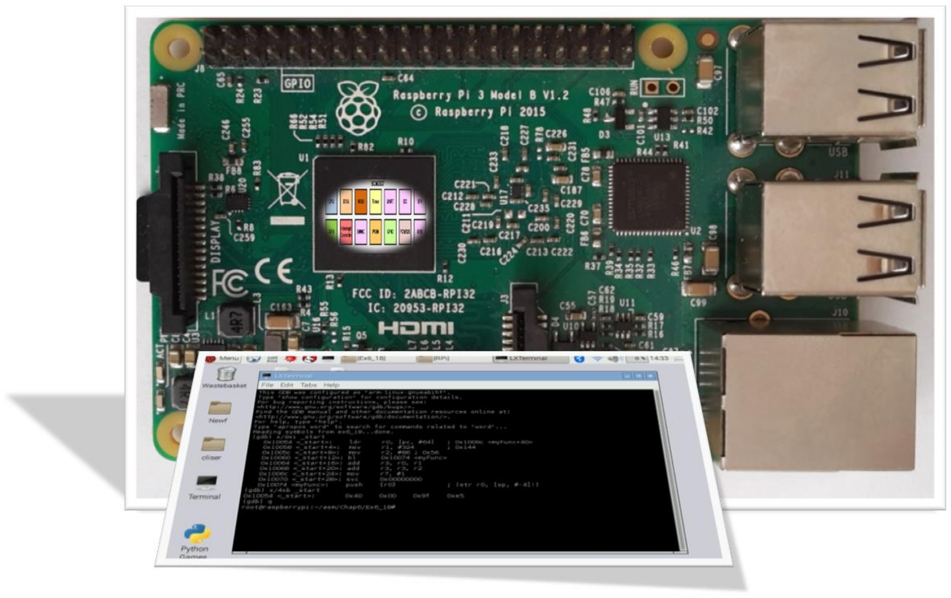
Raspberry Pi Assembly Programming using GCC

Step by Step Tutorial



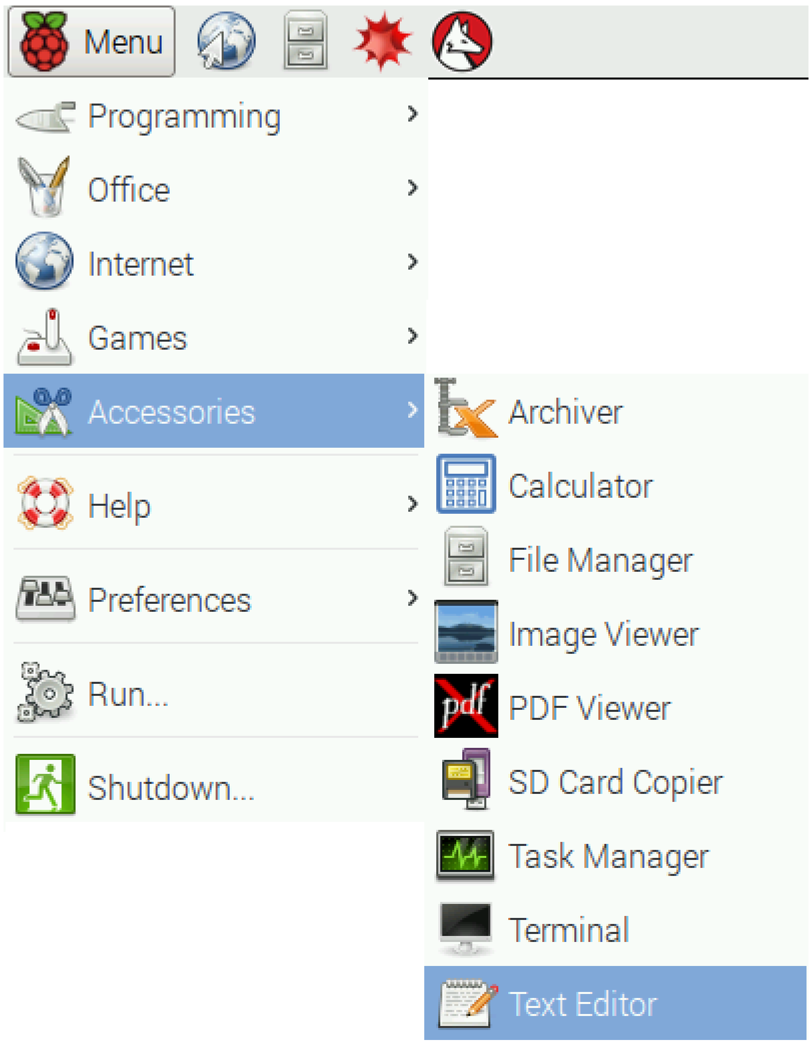
Azalia Yaghini

Sepehr Naimi

www.nicerLand.com

Writing a code

1. To write assembly programs, you need to use a text editor. You can use Leafpad which comes together with Raspbian. To open Leafpad, click on the *Menu* button, choose *accessories*, and then click on *Text* Editor.

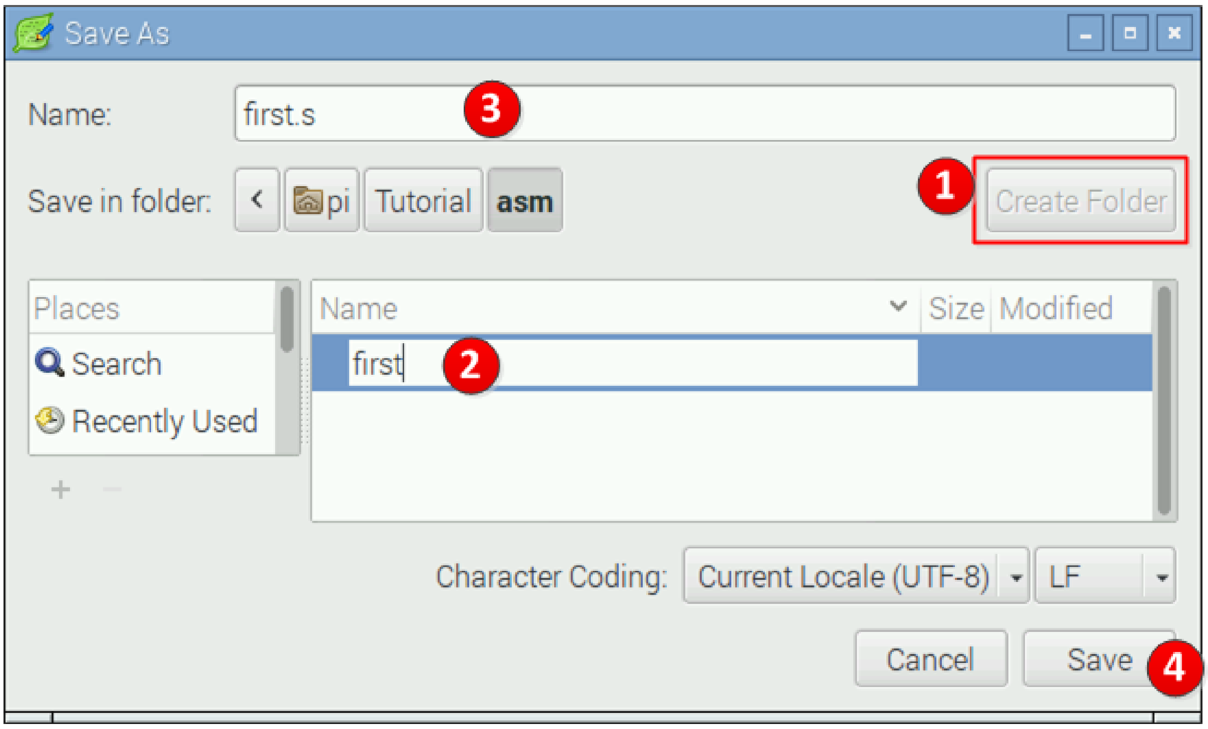


2. Type the following program in the editor:

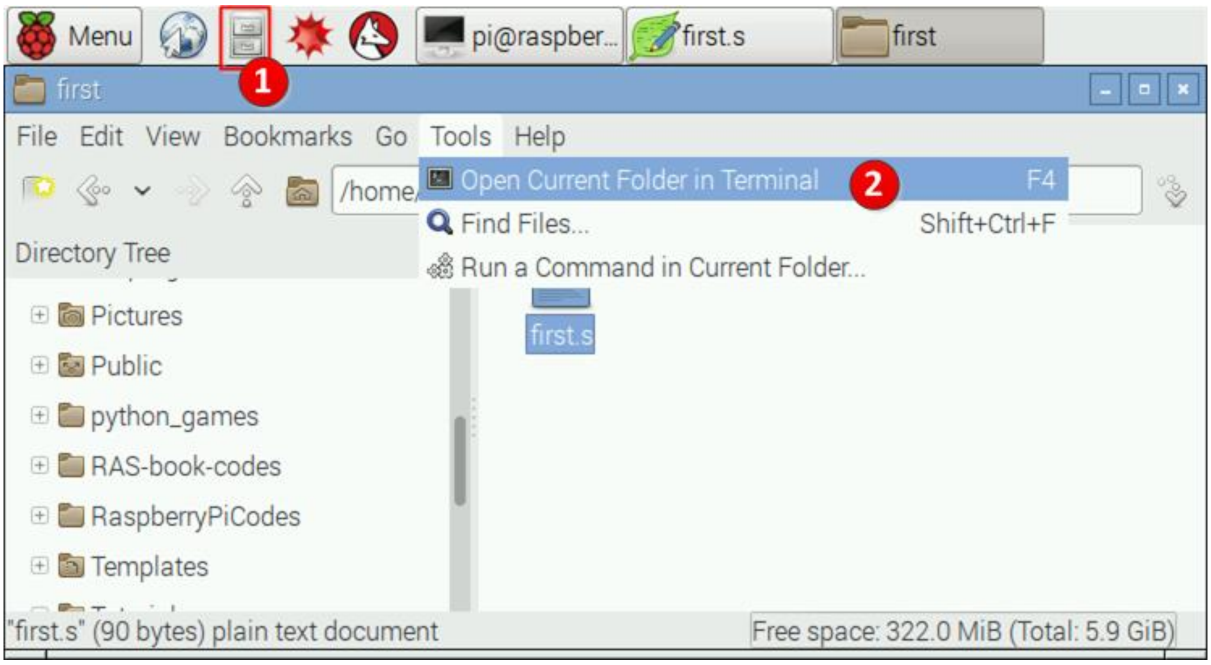
```
      .global _start
_start:
      mov    r1, #0x25
      mov    r2, #0x34
      add    r0, r1, r2

      mov    r7, #1
      svc    0
```

3. To save the program, press Ctrl+S. Make a directory for your assembly projects by pressing Create folder and name it *first and* press enter to go to the first directory. Then name your file as ***first.s*** and press the save button.



4. Open the first directory in the command prompt. To do so, open the *file manager* and go to your project directory which is named *first* in the case. Go to *Tools* on the menu bar, and click on Open Current Folder in Terminal (or press F4).



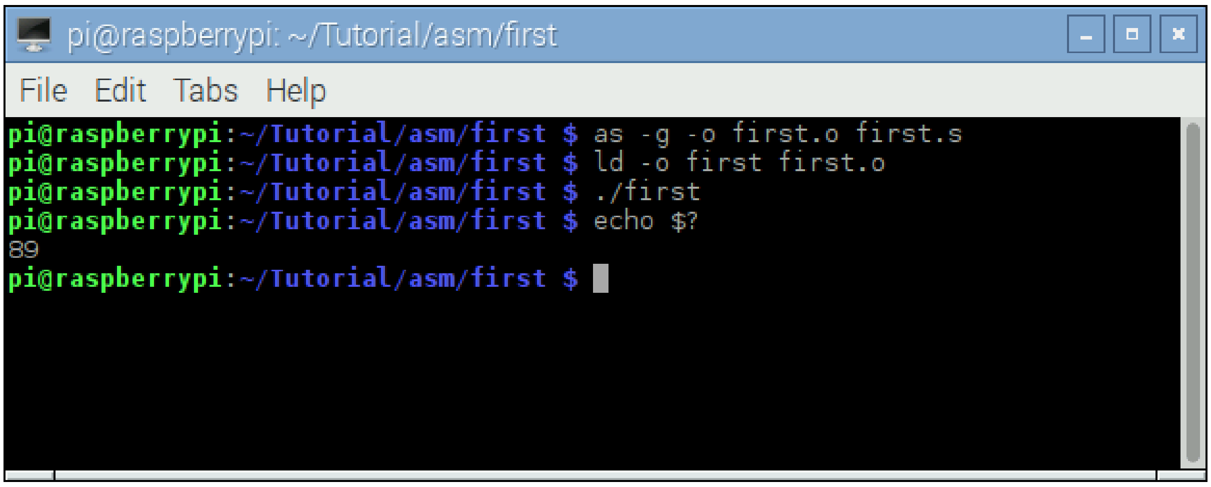
Assembling and Linking

5. To assemble the program, type the following in the command prompt.

```
as -g -o first.o first.s
```

6. Link the files by typing the following:

```
ld -o first first.o
```



```
pi@raspberrypi:~/Tutorial/asm/first $ as -g -o first.o first.s
pi@raspberrypi:~/Tutorial/asm/first $ ld -o first first.o
pi@raspberrypi:~/Tutorial/asm/first $ ./first
pi@raspberrypi:~/Tutorial/asm/first $ echo $?
89
pi@raspberrypi:~/Tutorial/asm/first $
```

Executing the program

7. To execute the program, type the following in the command prompt.

```
./first
```

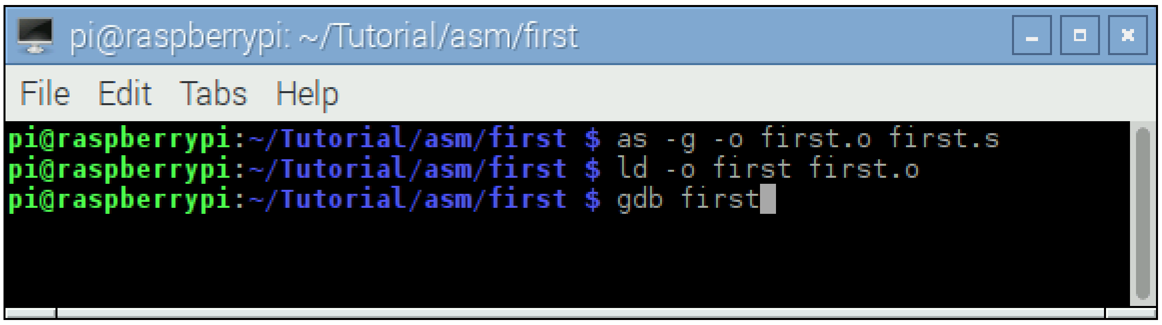
8. Programs can return a value through r0 register. The first program, stores the result of the add in r0. To see the return value, after running the program, type *echo $?*

```
echo $?
```

Debugging in GDB

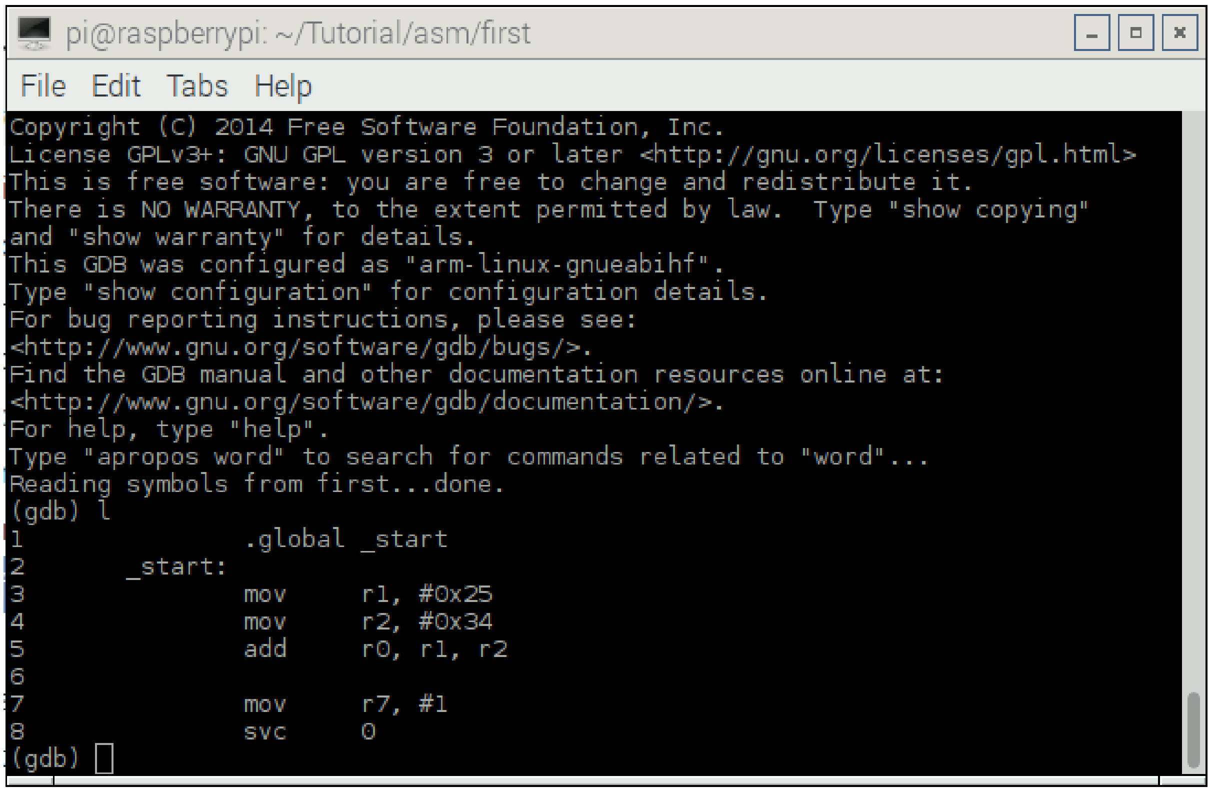
9. You can also debug your program in GDB. To do so, type the followings in the command prompt:

```
as -g -o first.o first.s
ld -o first first.o
gdb first
```



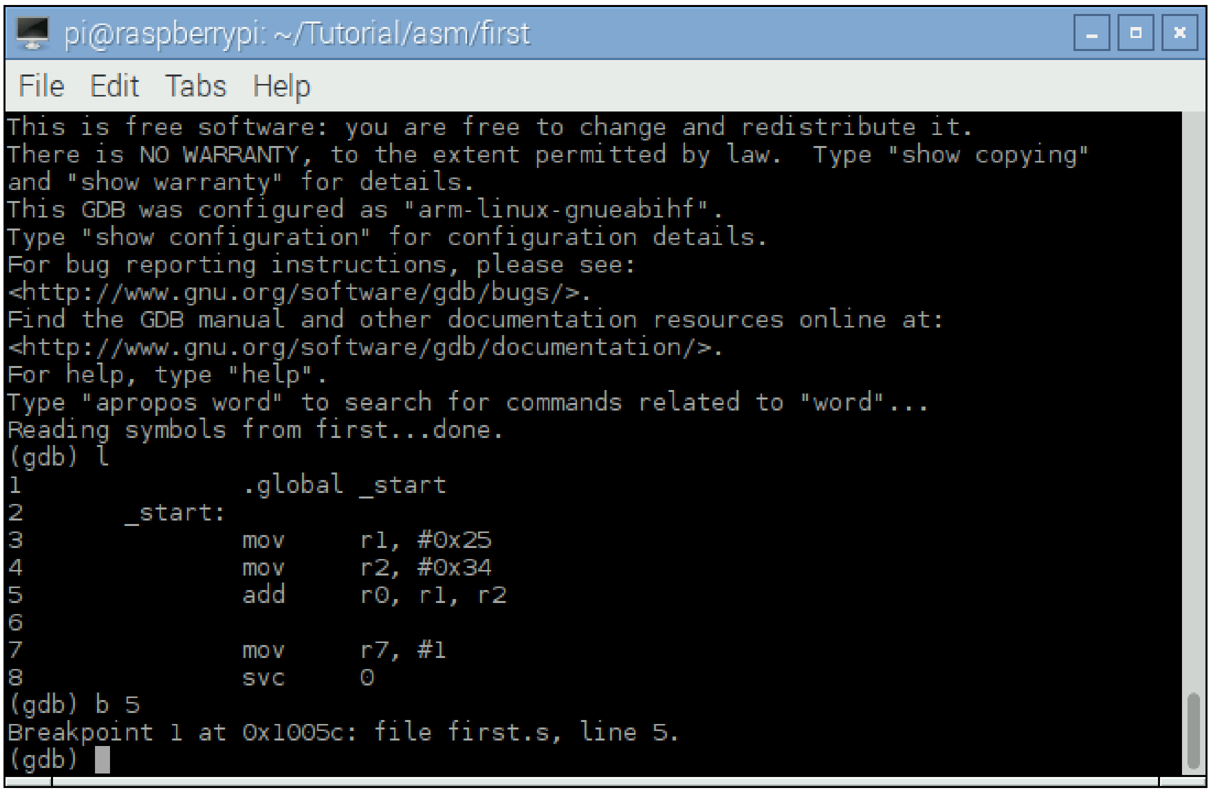
```
pi@raspberrypi:~/Tutorial/asm/first $ as -g -o first.o first.s
pi@raspberrypi:~/Tutorial/asm/first $ ld -o first first.o
pi@raspberrypi:~/Tutorial/asm/first $ gdb first
```

10. In gdb, type l and press enter to list the instructions of the program.



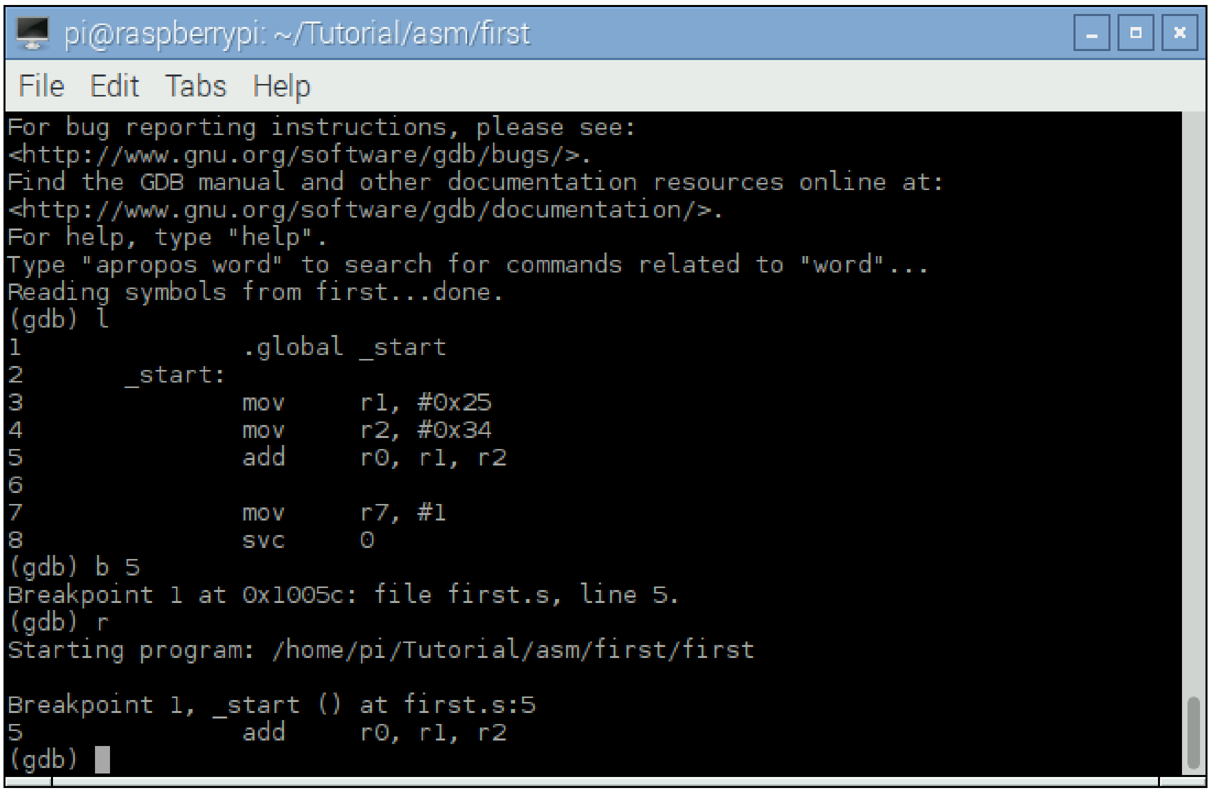
```
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "arm-linux-gnueabihf".
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 first...done.
(gdb) l
1               .global _start
2       _start:
3               mov     r1, #0x25
4               mov     r2, #0x34
5               add     r0, r1, r2
6
7               mov     r7, #1
8               svc     0
(gdb) 
```

11. Then type "b 5" to put break point on line 5. As the picture shows, the instruction of line 5 is located in address 0x1005C of memory. (It might be different in your system.)



```
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-gnueabihf".
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 first...done.
(gdb) l
1               .global _start
2       _start:
3               mov     r1, #0x25
4               mov     r2, #0x34
5               add     r0, r1, r2
6
7               mov     r7, #1
8               svc     0
(gdb) b 5
Breakpoint 1 at 0x1005c: file first.s, line 5.
(gdb) 
```

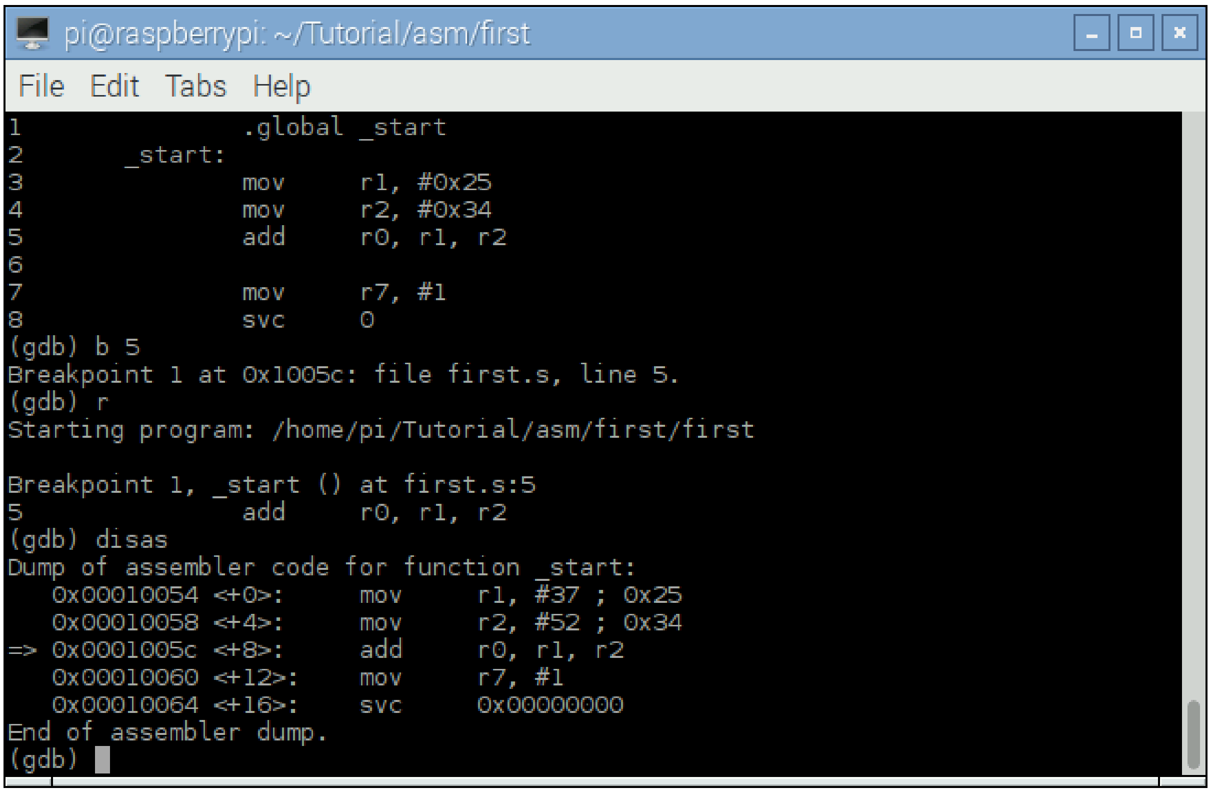
12. Type “r” to run the program to the break point.



```
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 first...done.
(gdb) l
1               .global _start
2       _start:
3               mov     r1, #0x25
4               mov     r2, #0x34
5               add     r0, r1, r2
6
7               mov     r7, #1
8               svc     0
(gdb) b 5
Breakpoint 1 at 0x1005c: file first.s, line 5.
(gdb) r
Starting program: /home/pi/Tutorial/asm/first/first

Breakpoint 1, _start () at first.s:5
5               add     r0, r1, r2
(gdb) 
```

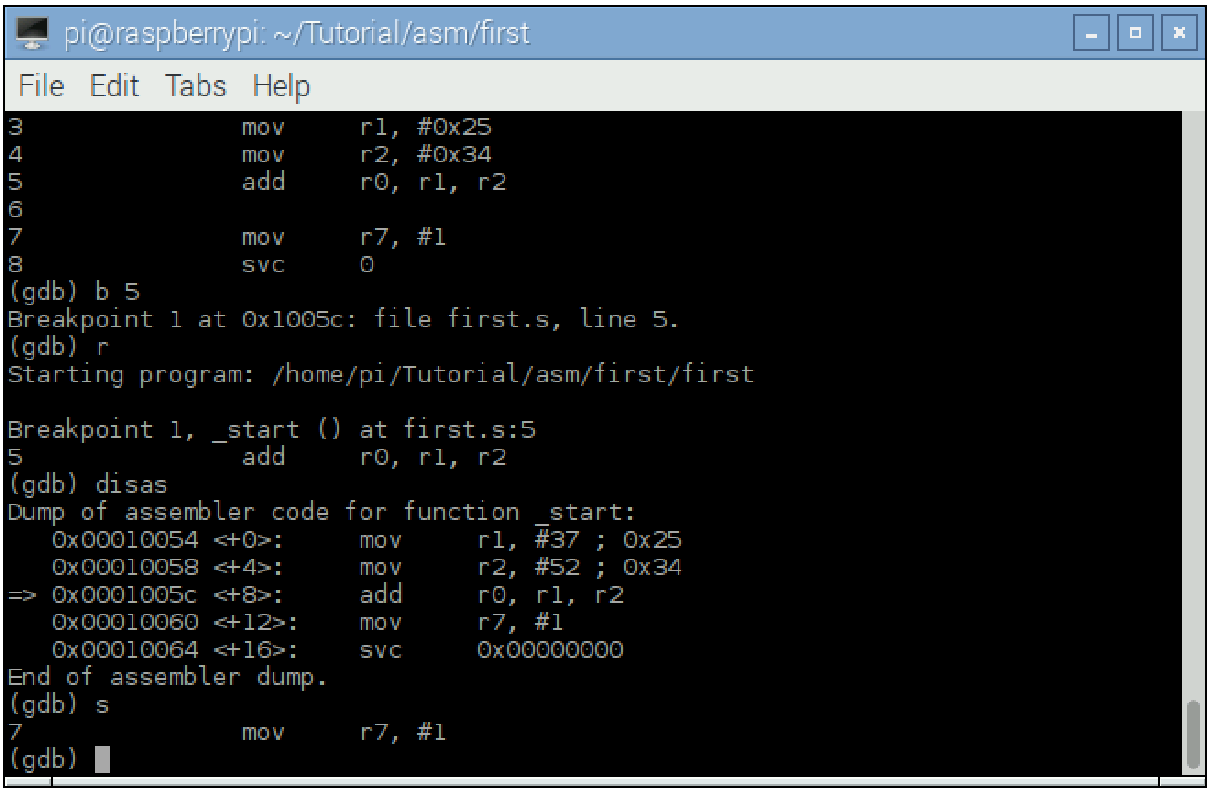
13. Type “disas” to disassemble your program. The disassemble shows the instructions together with their addresses in the memory. The next instruction to be executed in marked with “=>”.



```
1               .global _start
2       _start:
3               mov     r1, #0x25
4               mov     r2, #0x34
5               add     r0, r1, r2
6
7               mov     r7, #1
8               svc     0
(gdb) b 5
Breakpoint 1 at 0x1005c: file first.s, line 5.
(gdb) r
Starting program: /home/pi/Tutorial/asm/first/first

Breakpoint 1, _start () at first.s:5
5               add     r0, r1, r2
(gdb) disas
Dump of assembler code for function _start:
   0x00010054 <+0>:     mov     r1, #37 ; 0x25
   0x00010058 <+4>:     mov     r2, #52 ; 0x34
=> 0x0001005c <+8>:     add     r0, r1, r2
   0x00010060 <+12>:    mov     r7, #1
   0x00010064 <+16>:    svc     0x00000000
End of assembler dump.
(gdb) 
```

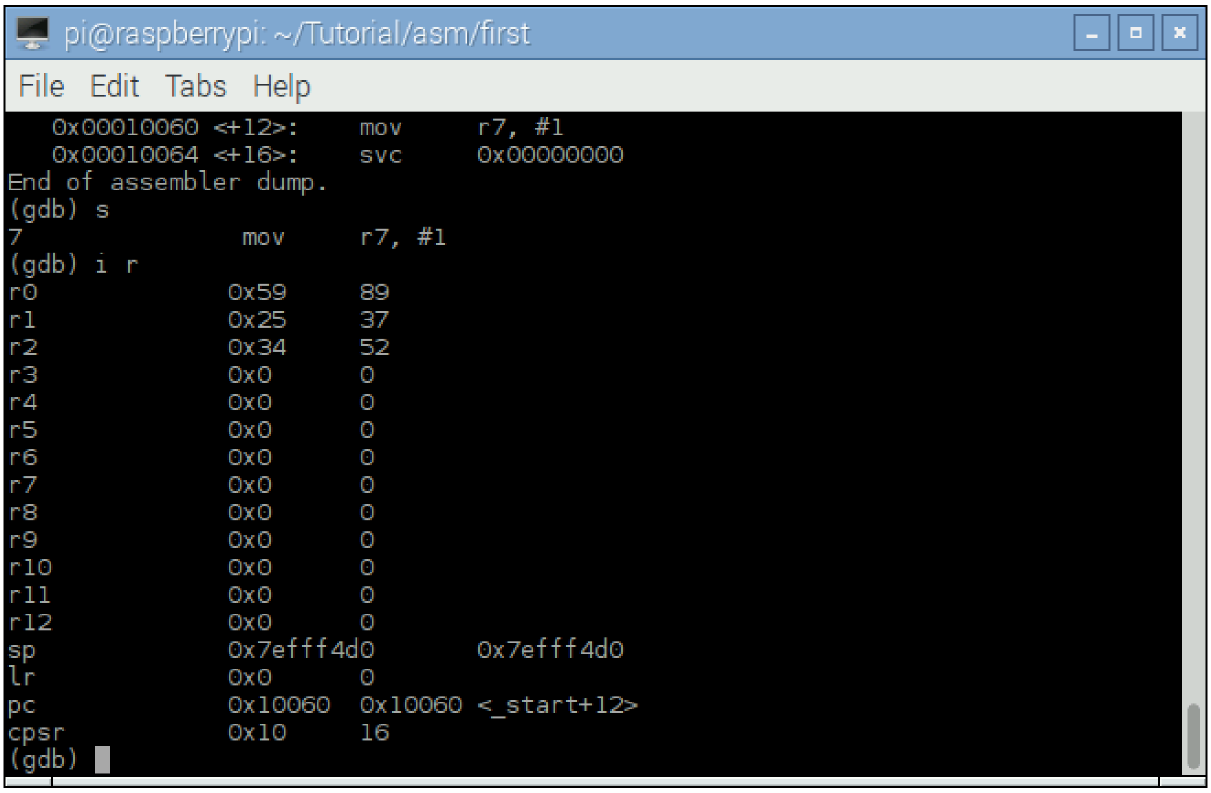
14. Type “s” to step the program. The step command executes the next instruction. So, the add instruction is executed in the case. As the picture shows, the next instruction to be executed is “mov r7, #1”.



```
3               mov     r1, #0x25
4               mov     r2, #0x34
5               add     r0, r1, r2
6
7               mov     r7, #1
8               svc     0
(gdb) b 5
Breakpoint 1 at 0x1005c: file first.s, line 5.
(gdb) r
Starting program: /home/pi/Tutorial/asm/first/first

Breakpoint 1, _start () at first.s:5
5               add     r0, r1, r2
(gdb) disas
Dump of assembler code for function _start:
   0x00010054 <+0>:     mov     r1, #37 ; 0x25
   0x00010058 <+4>:     mov     r2, #52 ; 0x34
=> 0x0001005c <+8>:     add     r0, r1, r2
   0x00010060 <+12>:    mov     r7, #1
   0x00010064 <+16>:    svc     0x00000000
End of assembler dump.
(gdb) s
7               mov     r7, #1
(gdb)
```

15. Type “i r” to monitor the values of the CPU registers. R0 contains



```
   0x00010060 <+12>:    mov     r7, #1
   0x00010064 <+16>:    svc     0x00000000
End of assembler dump.
(gdb) s
7               mov     r7, #1
(gdb) i r
r0             0x59     89
r1             0x25     37
r2             0x34     52
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             0x7efff4d0       0x7efff4d0
lr             0x0      0
pc             0x10060  0x10060 <_start+12>
cpsr           0x10     16
(gdb)
```

16. Type “q” to quit gdb.