

Full Variable register

length, %eax
width, %ebx
height, %ecx
gallons, %edx
rpm, %esi
horsepower, %edi
meters, %r8d
seconds, %r9d
mph, %r10d
x, %r11d

1) Using gdb to find variable values after _start

```
(gdb) stepi
0x00007ffff7fe40d1      45      in ./get-dynamic-info.h
1: /d $eax = 1879048185
2: /d $ebx = 1879047935
3: /d $ecx = -134227152
4: /d $edx = -134230144
(gdb) stepi
45      in ./get-dynamic-info.h
1: /d $eax = 1879048185
2: /d $ebx = 1879047935
3: /d $ecx = -134227152
4: /d $edx = -134230128
(gdb) stepi
0x00007ffff7fe40d8      45      in ./get-dynamic-info.h
1: /d $eax = 1879048185
2: /d $ebx = 1879047935
3: /d $ecx = -134227152
4: /d $edx = -134230128
(gdb) stepi
49      in ./get-dynamic-info.h
1: /d $eax = 1879048185
2: /d $ebx = 1879047935
3: /d $ecx = -134227152
4: /d $edx = -134230128
(gdb) █
```

2) “Hello World” to “hEllo World” after compiling “Hello World”

The screenshot shows a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with files like `hw3_HelloWorld.s`, `HW4_HelloWorld.s`, `HW4_GoZags.s`, and `AssemblyQuiz.s`. The code editor displays assembly code for `HW4_HelloWorld.s`, which is designed to print "hEllo World" by modifying the characters of the string "Hello World" in memory. The code uses `movb` to load and modify bytes at specific offsets from the `message` string. Comments explain the modifications: 'H' to 'h' at `message[0]` and 'e' to 'E' at `message[1]`. The code also sets up a `syscall` to write to `stdout` and includes an `Exit` section.

```

1  .global _start
2  .text
3  _start:
4      # Modify 'H' → 'h' at message[0]
5      movb    message(%rip), %al    # Load byte at message[0]
6      addb    $0x20, %al            # 'H' (0x48) + 0x20 = 'h' (0x68)
7      movb    %al, message(%rip)    # Store back
8
9      # Modify 'e' → 'E' at message[1]
10     movb    1+message(%rip), %al   # Load byte at message[1]
11     subb    $0x20, %al             # 'e' (0x65) - 0x20 = 'E' (0x45)
12     movb    %al, 1+message(%rip)   # Store back
13
14     mov     $1, %rax               # syscall: write
15     mov     $1, %rdi              # stdout
16     lea     message(%rip), %rsi    # address of string
17     mov     $13, %rdx             # string length
18     syscall
19
20     # Exit
21     mov     $60, %rax
22     xor     %rdi, %rdi
23     syscall
24
25 .section .data
26 message: .ascii "Hello World\n\0"

```

The terminal output shows the compilation and execution of the assembly code:

```

ndloan2@ada:~/MyFiles/260/homework4$ gcc -c HW4_HelloWorld.s
HW4_HelloWorld.s: Assembler messages:
HW4_HelloWorld.s: Warning: end of file not at end of a line; newline inserted
ndloan2@ada:~/MyFiles/260/homework4$ ld HW4_HelloWorld.o -o hello
ndloan2@ada:~/MyFiles/260/homework4$ ./hello
Hello World
ndloan2@ada:~/MyFiles/260/homework4$ gcc -c HW4_HelloWorld.s
HW4_HelloWorld.s: Assembler messages:
HW4_HelloWorld.s: Warning: end of file not at end of a line; newline inserted
ndloan2@ada:~/MyFiles/260/homework4$ ld HW4_HelloWorld.o -o hello
ndloan2@ada:~/MyFiles/260/homework4$ ./hello
hEllo World

```

3) “Go Zags!” to “GO ZAGS!” after compiling “Go Zags!”

```

1  .global _start
2  .text
3  _start:
4      lea    message(%rip), %rsi    # Load address of message into %rsi
5      mov    $0, %rcx              # index = 0
6
7
8  convert_loop:
9      movzbq (%rsi,%rcx,1), %rax    # Load byte at message[%rcx] into %rax
10     cmp    $0, %rax              # End of string?
11     je     done_convert
12
13     cmp    $'a', %al              # If char < 'a', skip
14     jl     skip
15     cmp    $'z', %al              # If char > 'z', skip
16     jg     skip
17
18     sub    $0x20, %al             # Convert to uppercase
19     mov    %al, (%rsi,%rcx,1)     # Store back
20
21 skip:
22     inc    %rcx
23     jmp    convert_loop
24
25 done_convert:
26     mov    $1, %rax              # syscall: write
27     mov    $1, %rdi              # stdout
28     lea    message(%rip), %rsi
29     mov    $9, %rdx              # message length
30     syscall
31
32     # Exit syscall
33     mov    $60, %rax
34     xor    %rdi, %rdi
35     syscall
36
37 .section .data
38 message: .ascii "Go Zags!\n"
39

```

```

ndean2@bada:~/files/260/homework4$ gcc -c HW4_GoZags.s
ndean2@bada:~/files/260/homework4$ ld HW4_GoZags.o -o zags
ndean2@bada:~/files/260/homework4$ ./zags
GO ZAGS!
ndean2@bada:~/files/260/homework4$

```