gcd function in sml, C, and asm

Saw Thinkar Nay Htoo

April 15, 2016

gcd

The greatest common divisor (gcd) of two positive natural numbers is the largest natural number that exactly divides both numbers. The gcd of 14 and 12 is 2, while the gcd of 14 and 11 is 1. The gcd is given by this specification:

$$\gcd : (\mathbb{N} \times \mathbb{N}) \to \mathbb{N}$$

$$\gcd(m, n) = \max\{d \in \mathbb{N} | m \bmod d = 0 \wedge n \bmod d = 0\}$$

One *algorithm* for calculating the gcd follows Euclid's method. If, for two positive natural numbers m and n, we have that $m > n$, then the gcd of m and n is defined by:

$$\text{euclid} : (\mathbb{N} \times \mathbb{N}) \to \mathbb{N}$$

$$\text{euclid}(m, n) = \begin{cases} \text{euclid}(n, m \bmod n), & \text{if } n > 0 \\ m, & \text{otherwise} \end{cases}$$

gcd in sml

This can be written in SML like this:

SML

```
fun euclid m n = if n > 0
                 then euclid n (m mod n)
                 else m ;
euclid 558 198; (* expect 18 *)
```

Although a short program, we may not be familiar with the use of recursion because it is not commonly used for C programs due to its inefficiency. But let's implement it that way anyway to follow the math definition more closely.

SML code result

```
> fun euclid m n = if n > 0
                   then euclid n (m mod n)
                   else m ;
euclid 558 198;  (* expect 18 *)
# # val euclid = fn: int -> int -> int
> val it = 18: int
```

gcd in c

C source code written to file lab3.c

```
#include <stdio.h>
int euclid(int m, int n)
{
  if( n > 0) return euclid(n, m % n);
  else return m;
}
int main()
{
  printf("GCD output = %i\n",euclid(558,198));
  return 0;
}
```

```
debian@debian:~/labs/lab3$ ./labc
GCD output = 18
```

The first line of this C code is an include to get access to the library. We need it here to use printf.

```
#include <stdio.h>
```

In ASM, we do not need to include any files in order to link to the C library, so we can skip this.

gcd in c: euclid function

```
int euclid(int m, int n)
{
  if( n > 0) return euclid(n, m % n);
  else return m;
}
```

These lines are the function definition. There are several techniques to be studied to implement it in assembler:

- syntax for function definitions
  C:
  `int euclid(){}`
  ASM: function is declared using "euclid:"
  `euclid:`

- parameter passing
  C:
  `int m, int n`
  ASM: variables are placed in stack.
  `push $198`
  `push $558`

- decision (if statement)
  C:
  `if() return euclid(); else return m;` ASM: the procedure will simply jump to endif.
  `jmp endif`
  `...`
  `endif:`

- conditional (relational expression to compare values)
  C:
  `if (n>0) return euclid()`
  `else reutrn m;`

ASM:eax is compared with zero. If eax is equal to zero, it will go to "else". If not it willl keep repeating the first function.

```
cmp $0, %eax
jle else
...
else:
...
```

- return statement
  C:
  `return`
  ASM: "ret" is simply used as "return"
  `ret`

- calling a function (recursively in this case)
  C:
  `euclid(558,198)`
  ASM: In ASM, function called using "call function-name"
  `call euclid`

- calculating modulus
  C:
  `m % n`
  ASM: two numbers are kept in the stack for one operation then placed in ebx and eax registers. edx is set to zero to clear off the previous value. idiv command is used to divide eax by, ebx giving out the remainder of edx. Click here for detailed explanation for idiv.

```
mov 12(%ebp), %ebx
mov  8(%ebp), %eax
mov $0, %edx
idiv %ebx
```

gcd in asm: main

The main function is simpler, but we need to also learn how to:

- call printf
- end the program

ASM code is put in the "text" section. The entry point is named "_start". It is a label (indicated by the colon). We make it global so the linker will make it visible to be called externally (by the operating system).

ASM source code written to file lab3.s

```
.text
.globl _start
_start:
```

printf needs 2 parameters: a format string and a value. That value must be determined by called our function euclid. Return values are found in the EAX register. The euclid function also requires 2 parameters which must be pushed onto the system stack so they can be retrieved within the function. Parameters are pushed right to left (the C convention). Immediate (literal) values are prefixed with the $ sign. Register names are prefixed with the % sign.

ASM source code appended to file lab3.s

```
push $198
push $558
call euclid
add $8, %esp #two of them 4 bytes each
.data
fmt: .string "GCD output = %d\n"
.text
push %eax
push $fmt
call printf
add $8, %esp
```

The program is ended by calling software interrupt 0x80. The 1 in EAX means exit command and the 0 in EBX is the convention to mean program had no errors.

ASM source code appended to file lab3.s

```
mov $1,%eax
mov $0,%ebx
int $0x80
```

gcd in asm: euclid function

The euclid function uses the same technique of creating a label to indicate start of function. It ends with the ret instruction. The first 2 instructions set up a stack frame base pointer (EBP) to give us access to paramters even if the stack pointer (ESP) moves.

ASM source code appended to file lab3.s

```
euclid:
push %ebp

```

Now we can access the 2 parameters using register EBP. 4 bytes above EBP is the return address, 8 is the 1st parm, and 12 is the second. i.e.

n is on stack at `12(%ebp)`

m is on stack at `8(%ebp)`

The if statement has to be simulated by branching to labels depending of results of doing the comparison of n to zero.

ASM source code appended to file lab3.s

```
mov %esp,%ebp
mov 12(%ebp),%eax
cmp $0,%eax
jle else
```

This is the "then" part of the if statement. We need to calculate $m \mod n$ and call euclid again! Integer division is done by putting dividend in EDX:EAX as 64-bit value, and divisor in EBX. The remainder will be in EDX.

ASM source code appended to file lab3.s

```
mov 12(%ebp), %ebx
mov 8(%ebp), %eax
mov $0, %edx # clear upper 32 bits of the 64-bit divident edx:eax
idiv %ebx # modulus is in edx (quotient is in eax)
push  %edx
push %ebx
call euclid
add $8,%esp
jmp endif
else:
mov 8(%ebp),%eax
endif:
```

These last 2 instructions undo the first 2–they restore the original stack as it was found on entry to the function.

ASM source code appended to file lab3.s

```
mov %ebp,%esp
pop %ebp
ret
```

Here is output from running the ASM program:

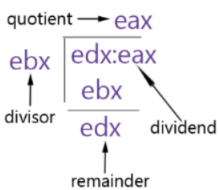
```
debian@debian:~/labs/lab3$ ./labasm
GCD output = 18
```

This is my own lab explanation.

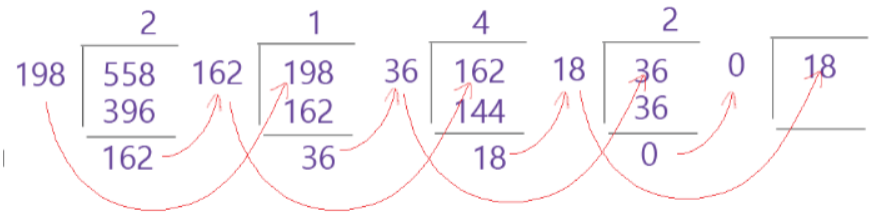
I made this lab report as simple as possible so that anyone without computing knowledge will be able to understand the content of this lab. In this lab, there are two most important things to understand fully: Euclidean Algorithm and Stack-frame.

What is Euclidean Algorithm

According to wikipedia, Euclidean algorithm is an efficient method for computing the greatest common divisor(gcd) of two numbers, the largest number that divides both of them without leaving a remainder. Below is the sample long division components compared with registers. Ok, what are those eax, ebx and edx? They are general purpose registers, where values are stored to do the calculation, and they have specific purposes. We will have to deal more with these registers when we write this algorithm using assembly language.



Lets say there are two numbers: 558 and 198. As we can see below, in the first step, dividend, 558 is divided by divisor, 198, resulting the remainder of 162. Then the remainder, 162 becomes the divisor in the second step, dividing the new dividend, 198 to give the new result of the remainder which is 36. The same procedure is repeated until the divisor becomes zero. So in this case, the GCD of 558 and 198 is 18.

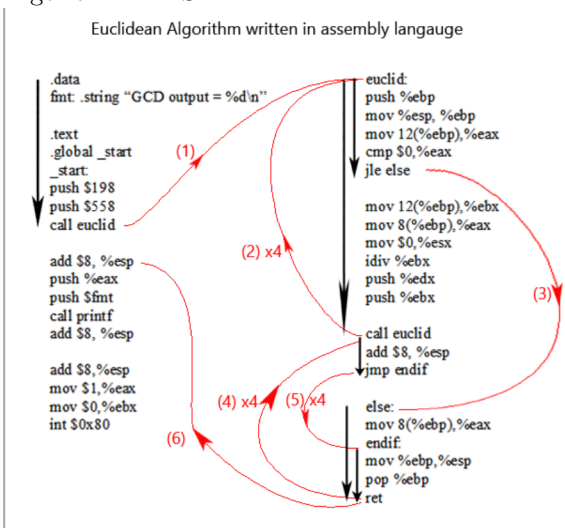


This link is the youtube video explanation for Euclidean Algorithm.
Now if you feel confident enough with your Euclidean Algorithm understanding, try to answer GCD for these numbers: (255,245), (531,234) and (126,186).
Using this online GCD calculator you can check your answer here.

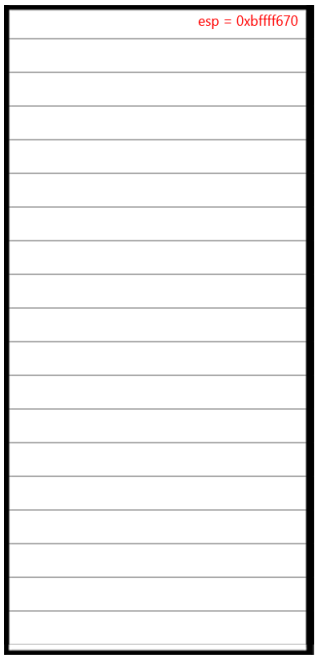
Click here for the Euclidean Algorithm in mathematical expression.
Click here for the alogrithm written in SML functional programming language.
Click here for the alogirthm written in C programming language.

Euclidean Algorithm in Assembly Language(ASM)

Now, lets begin with what assembly language is. It is a low-level programming language. This programming language is simple to write codes, if you understand the concept well, but you will have to write a lot just for a short program of high-level programming languages such as C, C++, and so on. Below is the Euclidean Algorithm in ASM.



The Stackframe: 1



This is the first time checking the stackframe before running the program.

```
(gdb) info r
eax            0x1c       28
ecx            0xb7fffc1c         -1207960548
edx            0xb7fedc90         -1208034160
ebx            0xb7fff000         -1207963648
esp            0xbffff670         0xbffff670
ebp            0x0        0x0
esi            0xbffff67c         -1073744260
edi            0x8048200          134513152
eip            0x8048200          0x8048200 <_start>
```

```
(gdb) x/8w $esp
0xbffff670:     1       -1073743937     0       -1073743907
0xbffff680:     -1073743889     -1073743871     -1073743855     -1073743844
```

```
(gdb) backtrace
#0  _start () at lab3.s:4
```

2

push $198

```
(gdb) info r
eax            0x1c     28
ecx            0xb7fffc1c       -1207960548
edx            0xb7fedc90       -1208034160
ebx            0xb7fff000       -1207963648
esp            0xbffff66c       0xbffff66c
ebp            0x0      0x0
esi            0xbffff67c       -1073744260
edi            0x8048200        134513152
eip            0x8048205        0x8048205 <_start+5>
(gdb) x/8w $esp
0xbffff66c:     198     1       -1073743937     0
0xbffff67c:     -1073743907     -1073743889     -1073743871     -1073743855
```

```
(gdb) backtrace
#0  _start () at lab3.s:5
```

3

push $558

```
(gdb) info r
eax            0x1c     28
ecx            0xb7fffc1c       -1207960548
edx            0xb7fedc90       -1208034160
ebx            0xb7fff000       -1207963648
esp            0xbffff668       0xbffff668
ebp            0x0      0x0
esi            0xbffff67c       -1073744260
edi            0x8048200        134513152
eip            0x804820a        0x804820a <_start+10>
```

```
(gdb) x/8w $esp
0xbffff668:     558     198     1       -1073743937
0xbffff678:     0       -1073743907     -1073743889     -1073743871
```

```
(gdb) backtrace
#0  _start () at lab3.s:6
```

4

call euclid

```
(gdb) info r
eax            0x1c     28
ecx            0xb7fffc1c       -1207960548
edx            0xb7fedc90       -1208034160
ebx            0xb7fff000       -1207963648
esp            0xbffff664       0xbffff664
ebp            0x0      0x0
esi            0xbffff67c       -1073744260
edi            0x8048200        134513152
eip            0x804822c        0x804822c <euclid>
(gdb) x/8w $esp
0xbffff664:     134513167       558     198     1
0xbffff674:     -1073743937     0       -1073743907     -1073743889
(gdb) backtrace
#0  euclid () at lab3.s:19
```

5

push %ebp

```
(gdb) info r
eax            0x1c       28
ecx            0xb7fffc1c         -1207960548
edx            0xb7fedc90         -1208034160
ebx            0xb7fff000         -1207963648
esp            0xbffff660         0xbffff660
ebp            0x0        0x0
esi            0xbffff67c         -1073744260
edi            0x8048200          134513152
eip            0x804822d          0x804822d <euclid+1>
```

```
(gdb) x/8w $esp
0xbffff660:     0       134513167       558     198
0xbffff670:     1       -1073743937     0       -1073743907
```

```
(gdb) backtrace
#0  euclid () at lab3.s:21
```

6

mov %esp,%ebp

```
(gdb) info r
eax            0x1c     28
ecx            0xb7fffc1c       -1207960548
edx            0xb7fedc90       -1208034160
ebx            0xb7fff000       -1207963648
esp            0xbffff660       0xbffff660
ebp            0xbffff660       0xbffff660
esi            0xbffff67c       -1073744260
edi            0x8048200        134513152
eip            0x804822f        0x804822f <euclid+3>
```

```
(gdb) x/8w $ebp
0xbffff660:     0       134513167       558     198
0xbffff670:     1       -1073743937     0       -1073743907
(gdb) x/8w $esp
0xbffff660:     0       134513167       558     198
0xbffff670:     1       -1073743937     0       -1073743907
```

```
(gdb) backtrace
#0  euclid () at lab3.s:22
#1  0x0804820f in _start () at lab3.s:6
```

7

mov 12(%ebp),%eax

```
(gdb) info r
eax            0xc6     198
ecx            0xb7fffc1c       -1207960548
edx            0xb7fedc90       -1208034160
ebx            0xb7fff000       -1207963648
esp            0xbffff660       0xbffff660
ebp            0xbffff660       0xbffff660
esi            0xbffff67c       -1073744260
edi            0x8048200        134513152
eip            0x8048232        0x8048232 <euclid+6>
(gdb) x/8w $ebp
0xbffff660:     0       134513167       558     198
0xbffff670:     1       -1073743937     0       -1073743907
(gdb) x/8w $esp
0xbffff660:     0       134513167       558     198
0xbffff670:     1       -1073743937     0       -1073743907
```

```
(gdb) backtrace
#0  euclid () at lab3.s:23
#1  0x0804820f in _start () at lab3.s:6
```

8

cmp $0,%eax

```
(gdb) info r
eax            0xc6     198
ecx            0xb7fffc1c       -1207960548
edx            0xb7fedc90       -1208034160
ebx            0xb7fff000       -1207963648
esp            0xbffff660       0xbffff660
ebp            0xbffff660       0xbffff660
esi            0xbffff67c       -1073744260
edi            0x8048200        134513152
eip            0x8048235        0x8048235 <euclid+9>
```

```
(gdb) x/8w $ebp
0xbffff660:     0       134513167       558     198
0xbffff670:     1       -1073743937     0       -1073743907
(gdb) x/8w $esp
0xbffff660:     0       134513167       558     198
0xbffff670:     1       -1073743937     0       -1073743907
```

```
(gdb) backtrace
#0  euclid () at lab3.s:24
#1  0x0804820f in _start () at lab3.s:6
```

9

jle else

```
(gdb) info r
eax            0xc6     198
ecx            0xb7fffc1c       -1207960548
edx            0xb7fedc90       -1208034160
ebx            0xb7fff000       -1207963648
esp            0xbffff660       0xbffff660
ebp            0xbffff660       0xbffff660
esi            0xbffff67c       -1073744260
edi            0x8048200        134513152
eip            0x8048237        0x8048237 <euclid+11>
```

```
(gdb) x/8w $esp
0xbffff660:     0       134513167       558     198
0xbffff670:     1       -1073743937     0       -1073743907
(gdb) x/8w $ebp
0xbffff660:     0       134513167       558     198
0xbffff670:     1       -1073743937     0       -1073743907
```

```
(gdb) backtrace
#0  euclid () at lab3.s:25
#1  0x0804820f in _start () at lab3.s:6
```

10

1	0xbffff670
198	0xbffff66c
558	esp = 0xbffff668

`mov 12(%ebp), %ebx`

```
(gdb) bt
#0  euclid () at lab3.s:26
#1  0x0804820f in _start () at lab3.s:6
(gdb) x/8w $esp
0xbffff660:     0       134513167       558     198
0xbffff670:     1       -1073743937     0       -1073743907
(gdb) x/8w $ebp
0xbffff660:     0       134513167       558     198
0xbffff670:     1       -1073743937     0       -1073743907
(gdb) info r
eax            0xc6     198
ecx            0xb7fffc1c       -1207960548
edx            0xb7fedc90       -1208034160
ebx            0xc6     198
esp            0xbffff660       0xbffff660
ebp            0xbffff660       0xbffff660
esi            0xbffff67c       -1073744260
edi            0x8048200        134513152
eip            0x804823a        0x804823a <euclid+14>
```

11

1	0xbffff670
198	0xbffff66c
558	esp = 0xbffff668

`mov 8(%ebp), %eax`

```
(gdb) bt
#0  euclid () at lab3.s:27
#1  0x0804820f in _start () at lab3.s:6
(gdb) x/8w $esp
0xbffff660:     0       134513167       558     198
0xbffff670:     1       -1073743937     0       -1073743907
(gdb) x/8w $ebp
0xbffff660:     0       134513167       558     198
0xbffff670:     1       -1073743937     0       -1073743907
(gdb) info r
eax            0x22e    558
ecx            0xb7fffc1c       -1207960548
edx            0xb7fedc90       -1208034160
ebx            0xc6     198
esp            0xbffff660       0xbffff660
ebp            0xbffff660       0xbffff660
esi            0xbffff67c       -1073744260
edi            0x8048200        134513152
eip            0x804823d        0x804823d <euclid+17>
```

121

mov 0, %edx

```
(gdb) bt
#0  euclid () at lab3.s:28
#1  0x0804820f in _start () at lab3.s:6
(gdb) x/8w $esp
0xbffff660:     0       134513167       558     198
0xbffff670:     1       -1073743937     0       -1073743907
(gdb) x/8w $ebp
0xbffff660:     0       134513167       558     198
0xbffff670:     1       -1073743937     0       -1073743907
(gdb) info r
eax            0x22e    558
ecx            0xb7fffc1c       -1207960548
edx            0x0      0
ebx            0xc6     198
esp            0xbffff660       0xbffff660
ebp            0xbffff660       0xbffff660
esi            0xbffff67c       -1073744260
edi            0x8048200        134513152
eip            0x8048242        0x8048242 <euclid+22>
```

13

idiv %ebx

```
(gdb) bt
#0  euclid () at lab3.s:29
#1  0x0804820f in _start () at lab3.s:6
(gdb) x/8w $esp
0xbffff660:     0       134513167       558     198
0xbffff670:     1       -1073743937     0       -1073743907
(gdb) x/8w $ebp
0xbffff660:     0       134513167       558     198
0xbffff670:     1       -1073743937     0       -1073743907
(gdb) info r
eax            0x2      2
ecx            0xb7fffc1c       -1207960548
edx            0xa2     162
ebx            0xc6     198
esp            0xbffff660       0xbffff660
ebp            0xbffff660       0xbffff660
esi            0xbffff67c       -1073744260
edi            0x8048200        134513152
eip            0x8048244        0x8048244 <euclid+24>
```

14

push %edx

```
(gdb) bt
#0  euclid () at lab3.s:30
#1  0x0804820f in _start () at lab3.s:6
(gdb) x/8w $esp
0xbffff65c:     162     0       134513167       558
0xbffff66c:     198     1       -1073743937     0
(gdb) x/8w $ebp
0xbffff660:     0       134513167       558     198
0xbffff670:     1       -1073743937     0       -1073743907
(gdb) info r
eax            0x2      2
ecx            0xb7fffc1c       -1207960548
edx            0xa2     162
ebx            0xc6     198
esp            0xbffff65c       0xbffff65c
ebp            0xbffff660       0xbffff660
esi            0xbffff67c       -1073744260
edi            0x8048200        134513152
eip            0x8048245        0x8048245 <euclid+25>
```

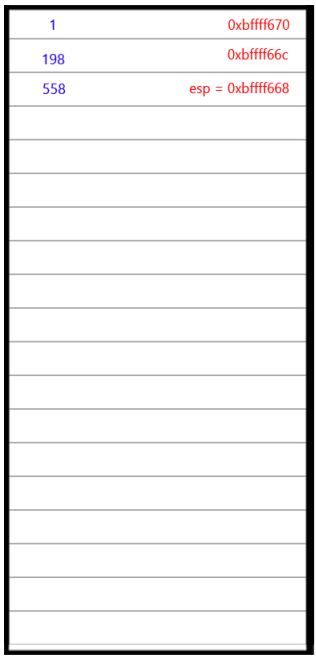
11

1	0xbffff670
198	0xbffff66c
558	esp = 0xbffff668

push %ebx

```
(gdb) bt
#0  euclid () at lab3.s:31
#1  0x0804820f in _start () at lab3.s:6
(gdb) x/8w $esp
0xbffff658:     198     162     0       134513167
0xbffff668:     558     198     1       -1073743937
(gdb) x/8w $ebp
0xbffff660:     0       134513167       558     198
0xbffff670:     1       -1073743937     0       -1073743907
(gdb) info r
eax            0x2      2
ecx            0xb7fffc1c       -1207960548
edx            0xa2     162
ebx            0xc6     198
esp            0xbffff658       0xbffff658
ebp            0xbffff660       0xbffff660
esi            0xbffff67c       -1073744260
edi            0x8048200        134513152
eip            0x8048246        0x8048246 <euclid+26>
```

16



`call euclid` There are a few things happened aftering calling the euclid function, and they are not obviously seen in the gdb. But by seeing the address changes we can say what happened there.

```
(gdb) bt
#0  euclid () at lab3.s:22
#1  0x0804824b in euclid () at lab3.s:31
#2  0x0804820f in _start () at lab3.s:6
(gdb) x/8w $esp
0xbffff650:     -1073744288     134513227       198     162
0xbffff660:     0       134513167       558     198
(gdb) x/8w $ebp
0xbffff650:     -1073744288     134513227       198     162
0xbffff660:     0       134513167       558     198
(gdb) info r
eax            0x2      2
ecx            0xb7fffc1c       -1207960548
edx            0xa2     162
ebx            0xc6     198
esp            0xbffff650       0xbffff650
ebp            0xbffff650       0xbffff650
esi            0xbffff67c       -1073744260
edi            0x8048200        134513152
eip            0x804822f        0x804822f <euclid+3>
```

17

1	0xbffff670
198	0xbffff66c
558	esp = 0xbffff668

mov 12(%ebp), %eax

```
(gdb) bt
#0  euclid () at lab3.s:23
#1  0x0804824b in euclid () at lab3.s:31
#2  0x0804820f in _start () at lab3.s:6
(gdb) x/8w $esp
0xbffff650:     -1073744288     134513227       198     162
0xbffff660:     0       134513167       558     198
(gdb) x/8w $ebp
0xbffff650:     -1073744288     134513227       198     162
0xbffff660:     0       134513167       558     198
(gdb) info r
eax            0xa2     162
ecx            0xb7fffc1c       -1207960548
edx            0xa2     162
ebx            0xc6     198
esp            0xbffff650       0xbffff650
ebp            0xbffff650       0xbffff650
esi            0xbffff67c       -1073744260
edi            0x8048200        134513152
eip            0x8048232        0x8048232 <euclid+6>
```

18

cmp $0,%eax

```
(gdb) bt
#0  euclid () at lab3.s:24
#1  0x0804824b in euclid () at lab3.s:31
#2  0x0804820f in _start () at lab3.s:6
(gdb) x/8w $esp
0xbffff650:     -1073744288     134513227       198     162
0xbffff660:     0       134513167       558     198
(gdb) x/8w $ebp
0xbffff650:     -1073744288     134513227       198     162
0xbffff660:     0       134513167       558     198
(gdb) info r
eax            0xa2     162
ecx            0xb7fffc1c       -1207960548
edx            0xa2     162
ebx            0xc6     198
esp            0xbffff650       0xbffff650
ebp            0xbffff650       0xbffff650
esi            0xbffff67c       -1073744260
edi            0x8048200        134513152
eip            0x8048235        0x8048235 <euclid+9>
```

19

jle else

```
(gdb) bt
#0  euclid () at lab3.s:25
#1  0x0804824b in euclid () at lab3.s:31
#2  0x0804820f in _start () at lab3.s:6
(gdb) x/8w $esp
0xbffff650:     -1073744288     134513227       198     162
0xbffff660:     0       134513167       558     198
(gdb) x/8w $ebp
0xbffff650:     -1073744288     134513227       198     162
0xbffff660:     0       134513167       558     198
(gdb) info r
eax            0xa2     162
ecx            0xb7fffc1c       -1207960548
edx            0xa2     162
ebx            0xc6     198
esp            0xbffff650       0xbffff650
ebp            0xbffff650       0xbffff650
esi            0xbffff67c       -1073744260
edi            0x8048200        134513152
eip            0x8048237        0x8048237 <euclid+11>
```

20

1	0xbffff670
198	0xbffff66c
558	esp = 0xbffff668

mov 12(%ebp), %ebx

```
(gdb) bt
#0  euclid () at lab3.s:26
#1  0x0804824b in euclid () at lab3.s:31
#2  0x0804820f in _start () at lab3.s:6
(gdb) x/8w $esp
0xbffff650:     -1073744288     134513227       198     162
0xbffff660:     0       134513167       558     198
(gdb) x/8w $ebp
0xbffff650:     -1073744288     134513227       198     162
0xbffff660:     0       134513167       558     198
(gdb) info r
eax            0xa2     162
ecx            0xb7fffc1c       -1207960548
edx            0xa2     162
ebx            0xa2     162
esp            0xbffff650       0xbffff650
ebp            0xbffff650       0xbffff650
esi            0xbffff67c       -1073744260
edi            0x8048200        134513152
eip            0x804823a        0x804823a <euclid+14>
```

21

1	0xbffff670
198	0xbffff66c
558	esp = 0xbffff668

mov 8(%ebp), %eax

```
(gdb) bt
#0  euclid () at lab3.s:27
#1  0x0804824b in euclid () at lab3.s:31
#2  0x0804820f in _start () at lab3.s:6
(gdb) x/8w $esp
0xbffff650:     -1073744288     134513227       198     162
0xbffff660:     0       134513167       558     198
(gdb) x/8w $ebp
0xbffff650:     -1073744288     134513227       198     162
0xbffff660:     0       134513167       558     198
(gdb) info r
eax            0xc6     198
ecx            0xb7fffc1c       -1207960548
edx            0xa2     162
ebx            0xa2     162
esp            0xbffff650       0xbffff650
ebp            0xbffff650       0xbffff650
esi            0xbffff67c       -1073744260
edi            0x8048200        134513152
eip            0x804823d        0x804823d <euclid+17>
```

22

mov $0, %edx

```
(gdb) bt
#0  euclid () at lab3.s:28
#1  0x0804824b in euclid () at lab3.s:31
#2  0x0804820f in _start () at lab3.s:6
(gdb) x/8w $esp
0xbffff650:     -1073744288     134513227       198     162
0xbffff660:     0       134513167       558     198
(gdb) x/8w $ebp
0xbffff650:     -1073744288     134513227       198     162
0xbffff660:     0       134513167       558     198
(gdb) info r
eax            0xc6     198
ecx            0xb7fffc1c       -1207960548
edx            0x0      0
ebx            0xa2     162
esp            0xbffff650       0xbffff650
ebp            0xbffff650       0xbffff650
esi            0xbffff67c       -1073744260
edi            0x8048200        134513152
eip            0x8048242        0x8048242 <euclid+22>
```

23

idiv %ebx

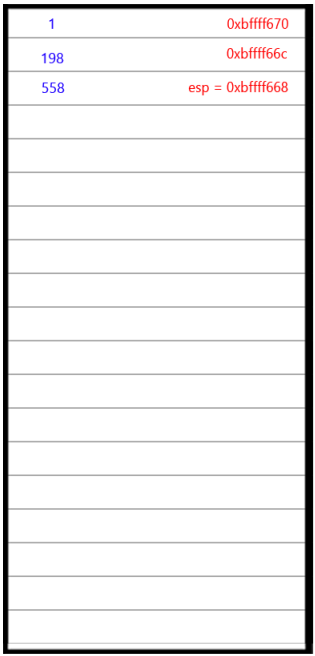
```
(gdb) bt
#0  euclid () at lab3.s:29
#1  0x0804824b in euclid () at lab3.s:31
#2  0x0804820f in _start () at lab3.s:6
(gdb) x/8w $esp
0xbffff650:     -1073744288     134513227       198     162
0xbffff660:     0       134513167       558     198
(gdb) x/8w $ebp
0xbffff650:     -1073744288     134513227       198     162
0xbffff660:     0       134513167       558     198
(gdb) info r
eax            0x1      1
ecx            0xb7fffc1c       -1207960548
edx            0x24     36
ebx            0xa2     162
esp            0xbffff650       0xbffff650
ebp            0xbffff650       0xbffff650
esi            0xbffff67c       -1073744260
edi            0x8048200        134513152
eip            0x8048244        0x8048244 <euclid+24>
```

24

push %edx

```
(gdb) bt
#0  euclid () at lab3.s:30
#1  0x0804824b in euclid () at lab3.s:31
#2  0x0804820f in _start () at lab3.s:6
(gdb) x/8w $esp
0xbffff64c:     36      -1073744288     134513227       198
0xbffff65c:     162     0       134513167       558
(gdb) x/8w $ebp
0xbffff650:     -1073744288     134513227       198     162
0xbffff660:     0       134513167       558     198
(gdb) info r
eax             0x1     1
ecx             0xb7fffc1c      -1207960548
edx             0x24    36
ebx             0xa2    162
esp             0xbffff64c      0xbffff64c
ebp             0xbffff650      0xbffff650
esi             0xbffff67c      -1073744260
edi             0x8048200       134513152
eip             0x8048245       0x8048245 <euclid+25>
```

25



push %ebx

```
(gdb) bt
#0  euclid () at lab3.s:31
#1  0x0804824b in euclid () at lab3.s:31
#2  0x0804820f in _start () at lab3.s:6
(gdb) x/8w $esp
0xbffff648:     162     36      -1073744288     134513227
0xbffff658:     198     162     0       134513167
(gdb) x/8w $ebp
0xbffff650:     -1073744288     134513227       198     162
0xbffff660:     0       134513167       558     198
(gdb) info r
eax             0x1     1
ecx             0xb7fffc1c      -1207960548
edx             0x24    36
ebx             0xa2    162
esp             0xbffff648      0xbffff648
ebp             0xbffff650      0xbffff650
esi             0xbffff67c      -1073744260
edi             0x8048200       134513152
eip             0x8048246       0x8048246 <euclid+26>
```

Text written to file labcode.sh

```
docsml lab3.doc
as -gstabs -o lab.o lab3.s
ld -dynamic-linker /lib/ld-linux.so.2 -o labasm lab.o -lc -lX11
gcc -Wall -o labc lab3.c
```

Text written to file labcode2.sh

```
gcc -Wall -o labc lab.c
gcc -Wall -o labasm lab.s
```

Text written to file labdoc.sh

```
doctex lab3.doc
pptexenv /home/debian/texfot.pl pdflatex lab3.tex
```

Bourne Shell

```
chmod 755 labcode2.sh
chmod 755 labcode.sh
chmod 755 labdoc.sh
```