CS 33: Computer Organization

Glenn Reinman
4731G Boelter Hall
reinman@cs.ucla.edu

Some notes adopted from Bryant and O'Hallaron

Course Components

- **Lectures**
  - Higher level concepts
- **Discussions**
  - Applied concepts, important tools and skills for labs, clarification of lectures, exam coverage
- **Labs**
  - The heart of the course
  - Provide in-depth understanding of an aspect of systems
  - Programming and measurement

More Info

- **Web**
  - Class web page hosted by CourseWeb
  - Copies of lectures, assignments, exams, solutions
  - Forum
- **Office Hours**
- **Textbook**
  - Randal E. Bryant and David R. O'Hallaron. "Computer Systems: A Programmer's Perspective", **3rd Edition**, Prentice Hall 2015.

Grading

- **Exams (55%)**
  - Midterm (20%)
  - Final (35%)
  - All exams are open book/open notes.
- **Labs (40%)**
  - 4 labs (10% each)
  - You must work alone on all labs
- **Homework (5%)**
  - 5 assignments (1% each)
  - Electronic submission only

Tentative Calendar

Week	Monday's	Wednesday's	Friday's
1	Intro + Bits and Bytes (1,2)	Integers (2)	Warmup Lab Due
2	Machine-Level Programming I: Basics (3)	Machine-Level Prog II: Control (3)	
3	Machine-Level Prog III: Procedures (3)	Machine-Level Prog IV: Data (3)	Data Lab Due
4	Machine-Level Prog V: Advanced Topics (3)	Floating Point (2)	
5	MIDTERM	Program Optimization (5)	Bomb Lab Due
6	The Memory Hierarchy (6)	Cache Memories (6)	
7	Concurrency (12+handouts)	Concurrency (12+handouts)	Attack Lab Due
8	Linking + Exceptions (7,8)	Virtual Memory (9)	
9	Holiday!	I/O (10)	
10	MIPS (handouts)	Review	Parallel Lab Due

- **Homework and Labs Due via CourseWeb by Midnight**

Final Exam: Thursday, June 11, 11:30 AM - 2:30 PM

Cheating

- **What is cheating?**
  - Sharing code: either by copying, retyping, looking at, or supplying a copy of a file.
- **What is NOT cheating?**
  - Helping others use systems or tools.
  - Helping others with high-level design issues.
  - Helping others debug their code.
- **Penalty for cheating:**
  - At the discretion of the Associate Dean

Lab Facilities

SEAS Administered Linux Machine

- `lnxsrv.seas.ucla.edu`
- Remote access only
  - Use ssh to log in with your SEAS account
- Please direct any account issues to the SEAS help desk as they are the only ones with root access on this machine

Alternatives (Not Recommended)

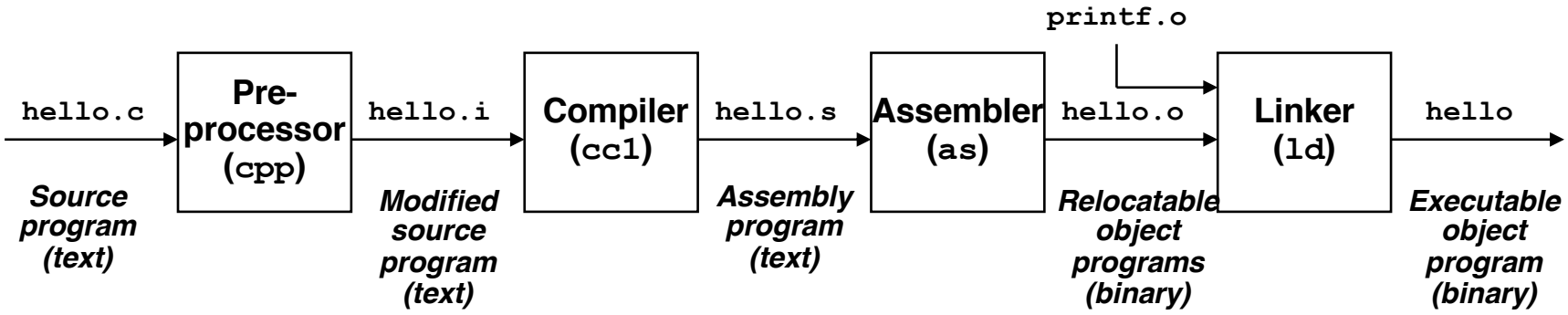
- You may use other alternatives to develop your code
- **BUT: We will test on the SEAS machines**
  - **Your code must work correctly on these machines for credit**

Course Theme

- **Abstraction is good, but don't forget reality!**
- **Abstractions have limits**
  - Things are more complex in hardware than they look in C/Java!!
  - Bugs are hard to track/understand if looking only from a high-level point of view
- **Useful outcomes**
  - Become more effective programmers
    - Able to find and eliminate bugs efficiently
    - Able to tune program performance
  - Prepare for later "systems" classes in CS
    - Compilers, Operating Systems, Networks, Computer Architecture, Parallel Programming

The Compilation System

```
#include <stdio.h>
int main()
{
    printf("hello, world\n");
}
```



Encoding Byte Values

- **Byte = 8 bits**
  - Binary 00000000_2 to 11111111_2
  - Decimal: 0_{10} to 255_{10}
  - Hexadecimal 00_{16} to FF_{16}
    - Base 16 number representation
    - Use characters ‘0’ to ‘9’ and ‘A’ to ‘F’
    - Write $FA1D37B_{16}$ in C as
      - 0xFA1D37B
      - 0xfa1d37b

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Bit-Level Operations in C

- **Operations &, |, ~, ^ Available in C**
  - Apply to any "integral" data type
    - `long, int, short, char, unsigned`
  - View arguments as bit vectors
  - Arguments applied bit-wise
- **Examples (Char data type)**
  - ~0x41 → 0xBE
    - $\sim 01000001_2 \rightarrow 10111110_2$
  - ~0x00 → 0xFF
    - $\sim 00000000_2 \rightarrow 11111111_2$
  - 0x69 & 0x55 → 0x41
    - $01101001_2 \;\&\; 01010101_2 \rightarrow 01000001_2$
  - 0x69 | 0x55 → 0x7D
    - $01101001_2 \mid 01010101_2 \rightarrow 01111101_2$

Contrast: Logic Operations in C

- **Contrast to Logical Operators**
  - &&, ||, !
    - View 0 as “False”
    - Anything nonzero as “True”
    - Always return 0 or 1
    - Early termination
- **Examples (char data type)**
  - !0x41 → 0x00
  - !0x00 → 0x01
  - !!0x41 → 0x01
  - 0x69 && 0x55 → 0x01
  - 0x69 || 0x55 → 0x01
  - p && *p (avoids null pointer access)

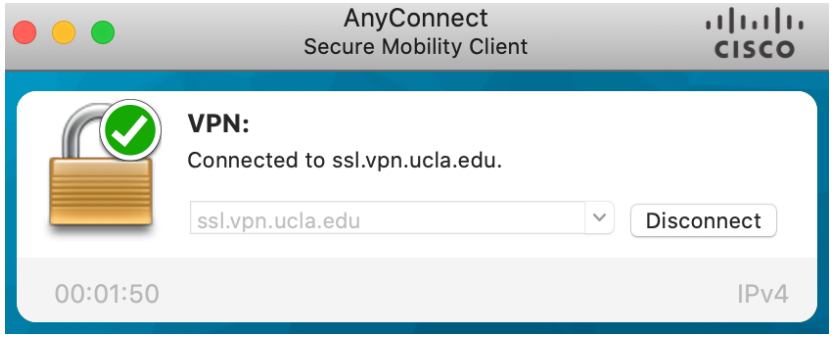
Shift Operations

- **Left Shift: `x << y`**
  - Shift bit-vector `x` left `y` positions
    - Throw away extra bits on left
  - Fill with 0's on right
- **Right Shift: `x >> y`**
  - Shift bit-vector `x` right `y` positions
    - Throw away extra bits on right
  - Logical shift
    - Fill with 0's on left
  - Arithmetic shift
    - Replicate most significant bit on left
- **Undefined Behavior**
  - Shift amount < 0 or ≥ word size

Argument x	01100010
<< 3	00010*000*
Log. >> 2	*00*011000
Arith. >> 2	*00*011000

Argument x	10100010
<< 3	00010*000*
Log. >> 2	*00*101000
Arith. >> 2	*11*101000

Connecting to Lab Computers




```
[reinman@Nyx ~ % ssh lnxsrv.seas.ucla.edu
[reinman@lnxsrv.seas.ucla.edu's password:
Last login: Sun Mar 29 22:00:26 2020 from 164.67.90.13
*******************************************************************************
                lnxsrv03.seas.ucla.edu RHEL 6
*******************************************************************************

    * Please use "lnxsrv.seas.ucla.edu" to login for load balancing
    * User processes older than 36 hours will be cleaned up
    * You can run graphical applications from your PC using SSH and Xming
    !! Please see http://www.seasnet.ucla.edu/UnixServers/lnxsrv !!

*******************************************************************************
*******************************************************************************
* SEASnet Computing Access                                                    *
*                                                                             *
* Priority is given both on the servers and in the student labs to those     *
* students doing coursework. Computing support for research is provided by   *
* each department.                                                            *
*******************************************************************************
* For assistance please contact help@seas.ucla.edu or call 206-6864.         *
*******************************************************************************
[reinman@lnxsrv03 ~]$
```

Editing

- Use your linux editor of choice (emacs, vi, …)

```
[[reinman@lnxsrv03 ~/code]$ emacs -nw bits-demo.c
```

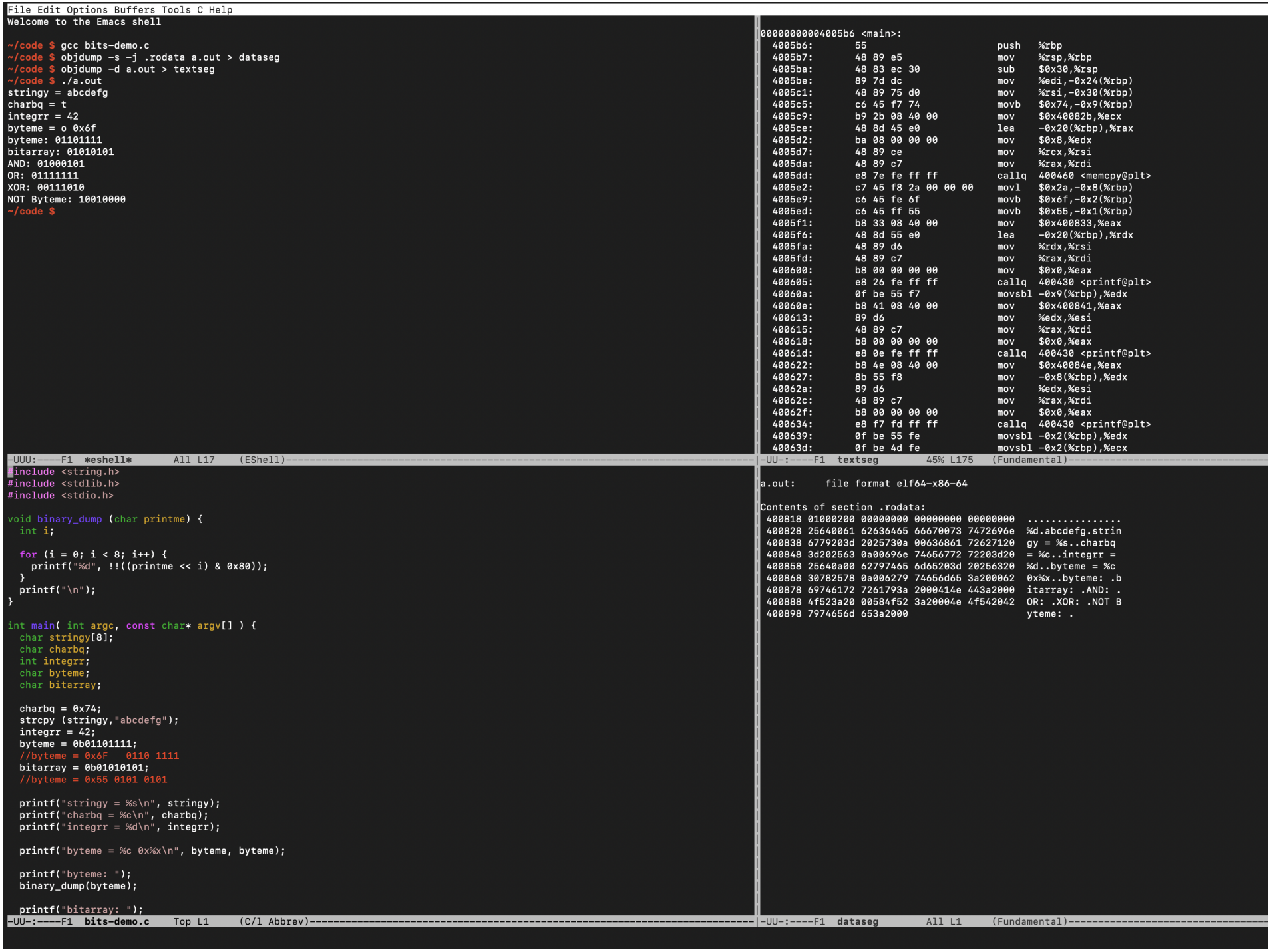
```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

void binary_dump (char printme) {
  int i;

  for (i = 0; i < 8; i++) {
    printf("%d", !!((printme << i) & 0x80));
  }
  printf("\n");
}

int main( int argc, const char* argv[] ) {
  char stringy[8];
  char charbq;
  int integrr;
  char byteme;
  char bitarray;

  charbq = 0x74;
  strcpy (stringy,"abcdefg");
  integrr = 42;
  byteme = 0b01101111;
  //byteme = 0x6F   0110 1111
  bitarray = 0b01010101;
  //byteme = 0x55 0101 0101

  printf("stringy = %s\n", stringy);
  printf("charbq = %c\n", charbq);
  printf("integrr = %d\n", integrr);

  printf("byteme = %c 0x%x\n", byteme, byteme);

  printf("byteme: ");
  binary_dump(byteme);

  printf("bitarray: ");
-UU-:----F1  bits-demo.c    Top L20    (C/l Abbrev)----------
```

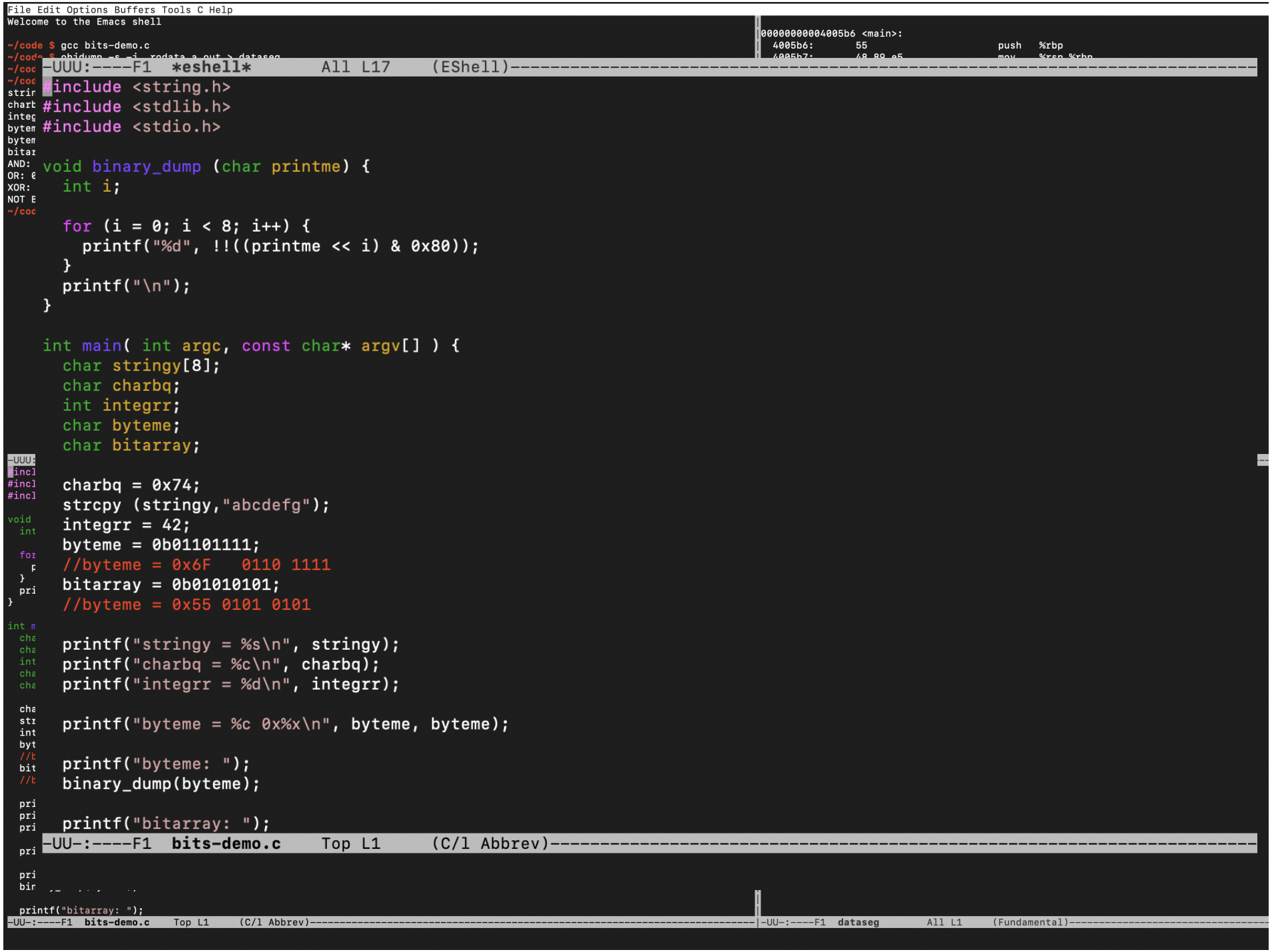
Sample C Code

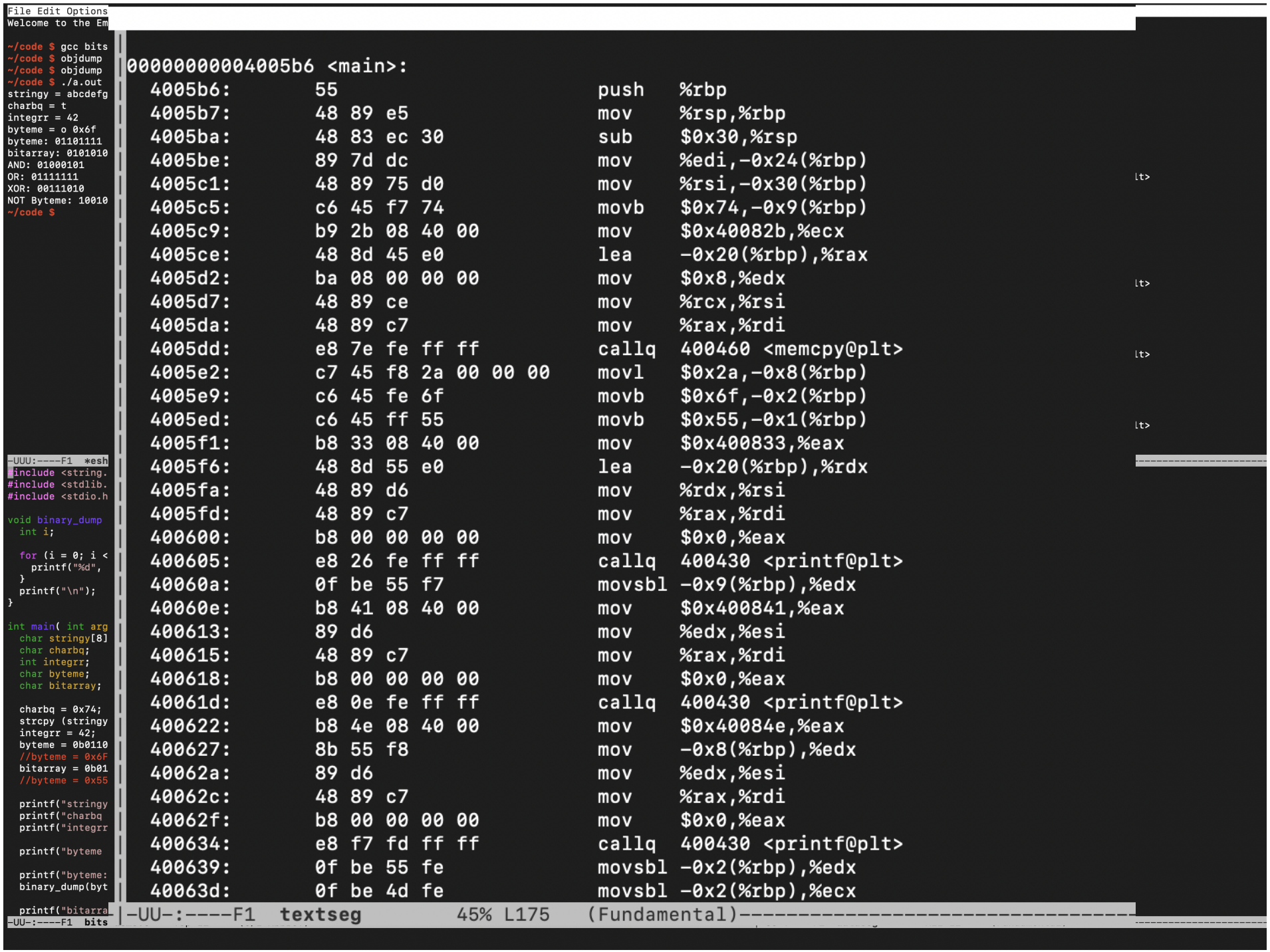
```
int main( int argc, const char* argv[] ) {
  char stringy[8];
  char charbq;
  int integrr;
  char byteme;
  char bitarray;

  charbq = 0x74;
  strcpy (stringy,"abcdefg");
  integrr = 42;
  byteme = 0b01101111;
  //byteme = 0x6F   0110 1111
  bitarray = 0b01010101;
  //byteme = 0x55 0101 0101

  printf("stringy = %s\n", stringy);
  printf("charbq = %c\n", charbq);
  printf("integrr = %d\n", integrr);

  printf("byteme = %c 0x%x\n", byteme, byteme);

  printf("byteme: ");
  binary_dump(byteme);

  printf("bitarray: ");
  binary_dump(bitarray);

  printf("AND: ");
  binary_dump(byteme & bitarray);

  printf("OR: ");
  binary_dump(byteme | bitarray);

  printf("XOR: ");
  binary_dump(byteme ^ bitarray);

  printf("NOT Byteme: ");
  binary_dump(~byteme);
-UU-:----F1  bits-demo.c    21% L33    (C/l Abbrev)----------
```

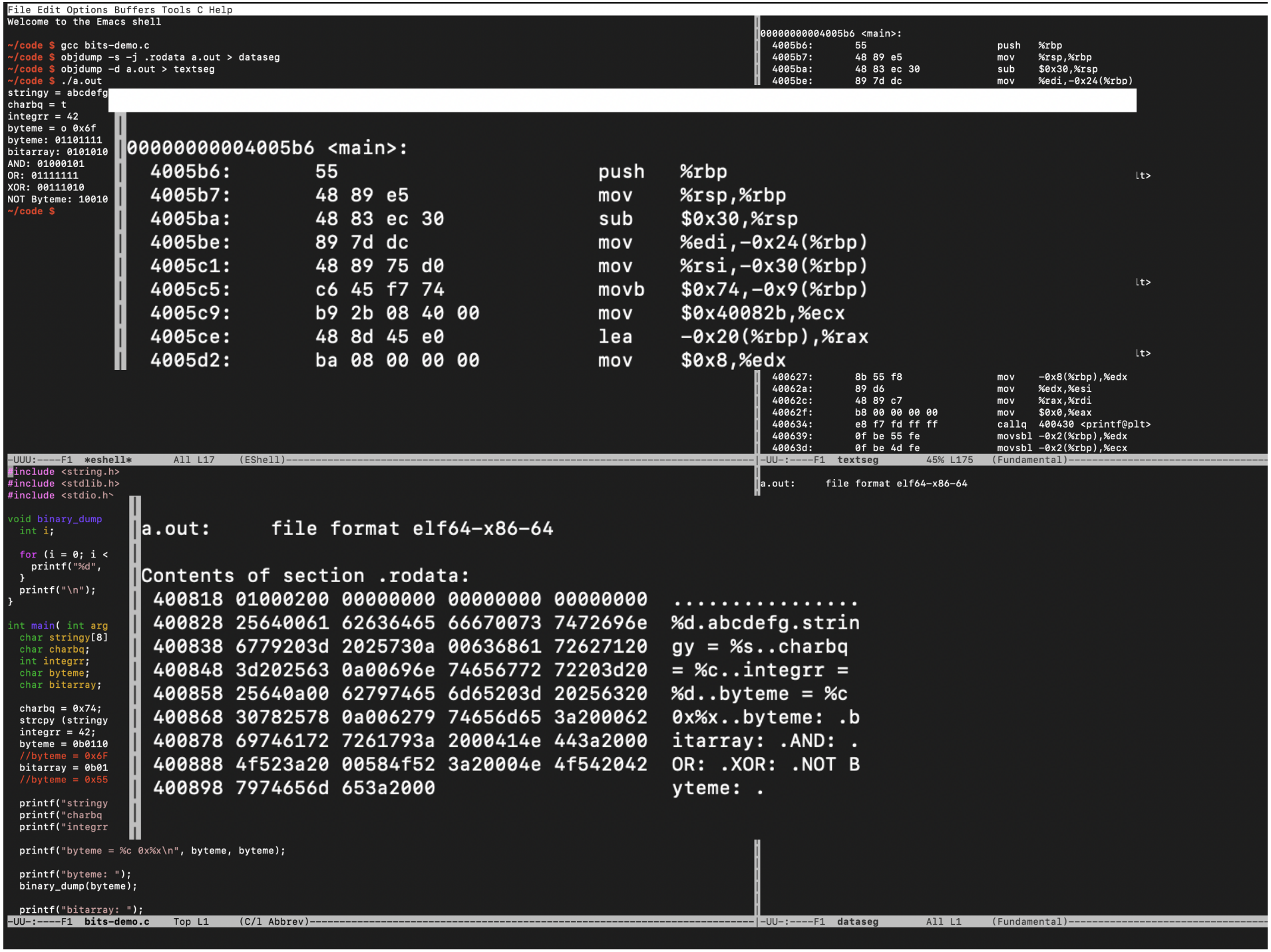
Code Interaction

```
Welcome to the Emacs shell

~/code $ gcc bits-demo.c
~/code $ objdump -s -j .rodata a.out > dataseg
~/code $ objdump -d a.out > textseg
~/code $ ./a.out
stringy = abcdefg
charbq = t
integrr = 42
byteme = o 0x6f
byteme: 01101111
bitarray: 01010101
AND: 01000101
OR: 01111111
XOR: 00111010
NOT Byteme: 10010000
~/code $
```


File Edit Options Buffers Tools C Help
Welcome to the Emacs shell
~/code $ gcc bits-demo.c
~/code $ objdump -s -j .rodata a.out > dataseg
~/code $ objdump -d a.out > textseg
~/code $ ./a.out
stringy = abcdefg
charbq = t
integrr = 42
byteme = o 0x6f
byteme: 01101111
bitarray: 01010101
AND: 01000101
OR: 01111111
XOR: 00111010
NOT Byteme: 10010000
~/code $
-UUU:----F1 *eshell* All L17 (EShell)
00000000004005b6 <main>:
4005b6: 55 push %rbp
4005b7: 48 89 e5 mov %rsp,%rbp
4005ba: 48 83 ec 30 sub $0x30,%rsp
4005be: 89 7d dc mov %edi,-0x24(%rbp)
4005c1: 48 89 75 d0 mov %rsi,-0x30(%rbp)
4005c5: c6 45 f7 74 movb $0x74,-0x9(%rbp)
4005c9: b9 2b 08 40 00 mov $0x40082b,%ecx
4005ce: 48 8d 45 e0 lea -0x20(%rbp),%rax
4005d2: ba 08 00 00 00 mov $0x8,%edx
4005d7: 48 89 ce mov %rcx,%rsi
4005da: 48 89 c7 mov %rax,%rdi
4005dd: e8 7e fe ff ff callq 400460 <memcpy@plt>
4005e2: c7 45 f8 2a 00 00 00 movl $0x2a,-0x8(%rbp)
4005e9: c6 45 fe 6f movb $0x6f,-0x2(%rbp)
4005ed: c6 45 ff 55 movb $0x55,-0x1(%rbp)
4005f1: b8 33 08 40 00 mov $0x400833,%eax
4005f6: 48 8d 55 e0 lea -0x20(%rbp),%rdx
4005fa: 48 89 d6 mov %rdx,%rsi
4005fd: 48 89 c7 mov %rax,%rdi
400600: b8 00 00 00 00 mov $0x0,%eax
400605: e8 26 fe ff ff callq 400430 <printf@plt>
40060a: 0f be 55 f7 movsbl -0x9(%rbp),%edx
40060e: b8 41 08 40 00 mov $0x400841,%eax
400613: 89 d6 mov %edx,%esi
400615: 48 89 c7 mov %rax,%rdi
400618: b8 00 00 00 00 mov $0x0,%eax
40061d: e8 0e fe ff ff callq 400430 <printf@plt>
400622: b8 4e 08 40 00 mov $0x40084e,%eax
400627: 8b 55 f8 mov -0x8(%rbp),%edx
40062a: 89 d6 mov %edx,%esi
40062c: 48 89 c7 mov %rax,%rdi
40062f: b8 00 00 00 00 mov $0x0,%eax
400634: e8 f7 fd ff ff callq 400430 <printf@plt>
400639: 0f be 55 fe movsbl -0x2(%rbp),%edx
40063d: 0f be 4d fe movsbl -0x2(%rbp),%ecx
-UU-:----F1 textseg 45% L175 (Fundamental)
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
void binary_dump (char printme) {
int i;
for (i = 0; i < 8; i++) {
printf("%d", !!((printme << i) & 0x80));
}
printf("\n");
}
int main(int argc, const char* argv[]) {
char stringy[8];
char charbq;
int integrr;
char byteme;
char bitarray;
charbq = 0x74;
strcpy (stringy,"abcdefg");
integrr = 42;
byteme = 0b01101111;
//byteme = 0x6F 0110 1111
bitarray = 0b01010101;
//byteme = 0x55 0101 0101
printf("stringy = %s\n", stringy);
printf("charbq = %c\n", charbq);
printf("integrr = %d\n", integrr);
printf("byteme = %c 0x%x\n", byteme, byteme);
printf("byteme: ");
binary_dump(byteme);
printf("bitarray: ");
-UU-:----F1 bits-demo.c Top L1 (C/l Abbrev)
a.out: file format elf64-x86-64
Contents of section .rodata:
400818 01000200 00000000 00000000 00000000 ....
400828 25640061 62636465 66670073 7472696e %d.abcdefg.strin
400838 6779203d 2025730a 00636861 72627120 gy = %s..charbq
400848 3d202563 0a00696e 74656772 72203d20 = %c..integrr =
400858 25640a00 62797465 6d65203d 20256320 %d..byteme = %c
400868 30782578 0a006279 74656d65 3a200062 0x%x..byteme: .b
400878 69746172 7261793a 2000414e 443a2000 itarray: .AND: .
400888 4f523a20 00584f52 3a20004e 4f542042 OR: .XOR: .NOT B
400898 7974656d 653a2000 yteme: .
-UU-:----F1 dataseg All L1 (Fundamental)


File Edit Options Buffers Tools C Help
Welcome to the Emacs shell
~/code $ gcc bits-demo.c
00000000004005b6 <main>:
4005b6: 55 push %rbp
-UUU:----F1 *eshell* All L17 (EShell)
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
void binary_dump (char printme) {
int i;
for (i = 0; i < 8; i++) {
printf("%d", !!((printme << i) & 0x80));
}
printf("\n");
}
int main(int argc, const char* argv[]) {
char stringy[8];
char charbq;
int integrr;
char byteme;
char bitarray;
charbq = 0x74;
strcpy (stringy,"abcdefg");
integrr = 42;
byteme = 0b01101111;
//byteme = 0x6F 0110 1111
bitarray = 0b01010101;
//byteme = 0x55 0101 0101
printf("stringy = %s\n", stringy);
printf("charbq = %c\n", charbq);
printf("integrr = %d\n", integrr);
printf("byteme = %c 0x%x\n", byteme, byteme);
printf("byteme: ");
binary_dump(byteme);
printf("bitarray: ");
-UU-:----F1 bits-demo.c Top L1 (C/l Abbrev)
printf("bitarray: ");
-UU-:----F1 bits-demo.c Top L1 (C/l Abbrev)
-UU-:----F1 dataseg All L1 (Fundamental)


File Edit Options
Welcome to the Em
~/code $ gcc bits
~/code $ objdump
~/code $ objdump
~/code $ ./a.out
stringy = abcdefg
charbq = t
integrr = 42
byteme = o 0x6f
byteme: 01101111
bitarray: 0101010
AND: 01000101
OR: 01111111
XOR: 00111010
NOT Byteme: 10010
~/code $
-UUU:----F1 *esh
#include <string.
#include <stdlib.
#include <stdio.h
void binary_dump
int i;
for (i = 0; i <
printf("%d",
}
printf("\n");
}
int main(int arg
char stringy[8]
char charbq;
int integrr;
char byteme;
char bitarray;
charbq = 0x74;
strcpy (stringy
integrr = 42;
byteme = 0b0110
//byteme = 0x6F
bitarray = 0b01
//byteme = 0x55
printf("stringy
printf("charbq
printf("integrr
printf("byteme
printf("byteme:
binary_dump(byt
printf("bitarra
-UU-:----F1 bits
00000000004005b6 <main>:
4005b6: 55 push %rbp
4005b7: 48 89 e5 mov %rsp,%rbp
4005ba: 48 83 ec 30 sub $0x30,%rsp
4005be: 89 7d dc mov %edi,-0x24(%rbp)
4005c1: 48 89 75 d0 mov %rsi,-0x30(%rbp)
4005c5: c6 45 f7 74 movb $0x74,-0x9(%rbp)
4005c9: b9 2b 08 40 00 mov $0x40082b,%ecx
4005ce: 48 8d 45 e0 lea -0x20(%rbp),%rax
4005d2: ba 08 00 00 00 mov $0x8,%edx
4005d7: 48 89 ce mov %rcx,%rsi
4005da: 48 89 c7 mov %rax,%rdi
4005dd: e8 7e fe ff ff callq 400460 <memcpy@plt>
4005e2: c7 45 f8 2a 00 00 00 movl $0x2a,-0x8(%rbp)
4005e9: c6 45 fe 6f movb $0x6f,-0x2(%rbp)
4005ed: c6 45 ff 55 movb $0x55,-0x1(%rbp)
4005f1: b8 33 08 40 00 mov $0x400833,%eax
4005f6: 48 8d 55 e0 lea -0x20(%rbp),%rdx
4005fa: 48 89 d6 mov %rdx,%rsi
4005fd: 48 89 c7 mov %rax,%rdi
400600: b8 00 00 00 00 mov $0x0,%eax
400605: e8 26 fe ff ff callq 400430 <printf@plt>
40060a: 0f be 55 f7 movsbl -0x9(%rbp),%edx
40060e: b8 41 08 40 00 mov $0x400841,%eax
400613: 89 d6 mov %edx,%esi
400615: 48 89 c7 mov %rax,%rdi
400618: b8 00 00 00 00 mov $0x0,%eax
40061d: e8 0e fe ff ff callq 400430 <printf@plt>
400622: b8 4e 08 40 00 mov $0x40084e,%eax
400627: 8b 55 f8 mov -0x8(%rbp),%edx
40062a: 89 d6 mov %edx,%esi
40062c: 48 89 c7 mov %rax,%rdi
40062f: b8 00 00 00 00 mov $0x0,%eax
400634: e8 f7 fd ff ff callq 400430 <printf@plt>
400639: 0f be 55 fe movsbl -0x2(%rbp),%edx
40063d: 0f be 4d fe movsbl -0x2(%rbp),%ecx
-UU-:----F1 textseg 45% L175 (Fundamental)


File Edit Options Buffers Tools C Help
Welcome to the Emacs shell
~/code $ gcc bits-demo.c
~/code $ objdump -s -j .rodata a.out > dataseg
~/code $ objdump -d a.out > textseg
~/code $ ./a.out
stringy = abcdefg
charbq = t
integrr = 42
byteme = o 0x6f
byteme: 01101111
bitarray: 0101010
AND: 01000101
OR: 01111111
XOR: 00111010
NOT Byteme: 10010
~/code $
00000000004005b6 <main>:
4005b6: 55 push %rbp
4005b7: 48 89 e5 mov %rsp,%rbp
4005ba: 48 83 ec 30 sub $0x30,%rsp
4005be: 89 7d dc mov %edi,-0x24(%rbp)
4005c1: 48 89 75 d0 mov %rsi,-0x30(%rbp)
4005c5: c6 45 f7 74 movb $0x74,-0x9(%rbp)
4005c9: b9 2b 08 40 00 mov $0x40082b,%ecx
4005ce: 48 8d 45 e0 lea -0x20(%rbp),%rax
4005d2: ba 08 00 00 00 mov $0x8,%edx
400627: 8b 55 f8 mov -0x8(%rbp),%edx
40062a: 89 d6 mov %edx,%esi
40062c: 48 89 c7 mov %rax,%rdi
40062f: b8 00 00 00 00 mov $0x0,%eax
400634: e8 f7 fd ff ff callq 400430 <printf@plt>
400639: 0f be 55 fe movsbl -0x2(%rbp),%edx
40063d: 0f be 4d fe movsbl -0x2(%rbp),%ecx
-UUU:----F1 *eshell* All L17 (EShell)
-UUU:----F1 textseg 45% L175 (Fundamental)
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
void binary_dump
int i;
for (i = 0; i <
printf("%d",
}
printf("\n");
}
int main(int arg
char stringy[8]
char charbq;
int integrr;
char byteme;
char bitarray;
charbq = 0x74;
strcpy (stringy
integrr = 42;
byteme = 0b0110
//byteme = 0x6F
bitarray = 0b01
//byteme = 0x55
printf("stringy
printf("charbq
printf("integrr
printf("byteme = %c 0x%x\n", byteme, byteme);
printf("byteme: ");
binary_dump(byteme);
printf("bitarray: ");
a.out: file format elf64-x86-64
Contents of section .rodata:
400818 01000200 00000000 00000000 00000000 ....
400828 25640061 62636465 66670073 7472696e %d.abcdefg.strin
400838 6779203d 2025730a 00636861 72627120 gy = %s..charbq
400848 3d202563 0a00696e 74656772 72203d20 = %c..integrr =
400858 25640a00 62797465 6d65203d 20256320 %d..byteme = %c
400868 30782578 0a006279 74656d65 3a200062 0x%x..byteme: .b
400878 69746172 7261793a 2000414e 443a2000 itarray: .AND: .
400888 4f523a20 00584f52 3a20004e 4f542042 OR: .XOR: .NOT B
400898 7974656d 653a2000 yteme: .
-UU-:----F1 bits-demo.c Top L1 (C/l Abbrev)
-UU-:----F1 dataseg All L1 (Fundamental)

GDB – GNU Debugger

```
~/code $ gdb ./a.out
GNU gdb (GDB) Red Hat Enterprise Linux (7.2-92.el6)
Copyright (C) 2010 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 "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /w/fac.3/cs/reinman/code/a.out...(no debugging symbols found)...done.
(gdb) break *0x400605
Breakpoint 1 at 0x400605
(gdb) run
Starting program: /w/fac.3/cs/reinman/code/a.out

Breakpoint 1, 0x0000000000400605 in main ()
Missing separate debuginfos, use: debuginfo-install glibc-2.12-1.212.el6_10.3.x86_64
(gdb)
```

```
(gdb) i r
rax            0x0        0
rbx            0x0        0
rcx            0x67666564636261
rdx            0x7fffffffe810
rsi            0x7fffffffe810
rdi            0x400833 4196403
rbp            0x7fffffffe830
rsp            0x7fffffffe800
r8             0x3e8c18fba0
r9             0x3e8ba0ee20
r10            0x7fffffffe580
r11            0x3e8be89720
r12            0x400470 4195440
r13            0x7fffffffe910
r14            0x0        0
r15            0x0        0
rip            0x400605 0x400605
eflags         0x246      [ PF ZF
cs             0x33       51
ss             0x2b       43
ds             0x0        0
es             0x0        0
fs             0x0        0
gs             0x0        0
(gdb) x/64xb $rsi
0x7fffffffe810: 0x61    0x62    0x63    0x64    0x65    0x66    0x67    0x00
0x7fffffffe818: 0x70    0x04    0x40    0x00    0x00    0x00    0x00    0x00
0x7fffffffe820: 0x10    0xe9    0xff    0xff    0xff    0x7f    0x00    0x74
0x7fffffffe828: 0x2a    0x00    0x00    0x00    0x00    0x00    0x6f    0x55
0x7fffffffe830: 0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x7fffffffe838: 0x20    0xed    0xe1    0x8b    0x3e    0x00    0x00    0x00
0x7fffffffe840: 0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x7fffffffe848: 0x18    0xe9    0xff    0xff    0xff    0x7f    0x00    0x00
(gdb)
-UUU:**--F1  *eshell*       Bot L69    (EShell)----------
```

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Lab 0

```
                --
/*
 * ezThreeFourths - multiplies by 3/4 rounding toward 0,
 *   Should exactly duplicate effect of C expression (x*3/4),
 *   including overflow behavior.
 *   Examples: ezThreeFourths(11) = 8
 *             ezThreeFourths(-9) = -6
 *             ezThreeFourths(1073741824) = -268435456 (overflow)
 *   Legal ops: ! ~ & ^ | + << >>
 *   Max ops: 12
 *   Rating: 3
 */
int ezThreeFourths(int x) {
```