COMPUTER SYSTEMS AND ORGANIZATION

C compilation

Daniel G. Graham Ph.D

UNIVERSITY of VIRGINIA | ENGINEERING

CMP INSTRUCTIONS

cmp A , B

jx

B op A

J[op]

Notice order is swapped

B > A

jg

B <= A

jle

B < A

jl

ESCAPE ROOM FUN

```
escapeRoom:
  leal (%rdi,%rdi), %eax
  cmpl $5, %eax
  jg .L3
  cmpl $1, %edi
  jne .L4
  movl $1, %eax
  ret
.L3:
  movl $1, %eax
  ret
.L4:
  movl $0, %eax
  ret
```

What must be passed to the Escape Room so that it returns true. Assume that we can supply an integer as input.

ESCAPE ROOM FUN

```
escapeRoom:
  leal (%rdi,%rdi), %eax
  cmpl $5, %eax
  jg .L3
  cmpl $1, %edi
  jne .L4
  movl $1, %eax
  ret
.L3:
  movl $1, %eax
  ret
.L4:
  movl $0, %eax
  ret
```

What must be passed to the Escape Room so that it returns true

First param > 2 or == 1

C MAIN ENTRY

```c
#include <stdio.h>

int main(void)
{
   puts("Hello World");
   return 0;
}
```

What is this return 0;

It is a status code.

C MAIN ENTRY

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

int main(void)
{
   printf("Hello World\n");
   return EXIT_SUCCESS;
}
```

WHEN WOULD WE USE STATUS CODE

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
  if (puts("Hello, world!") == EOF) {
    return EXIT_FAILURE;
    // code here never executes
  }
  return EXIT_SUCCESS;
  // code here never executes
}
```

(Let's do a quick demo of the manual/info page. Looking up a couple of things.

- Point out return value
- Know bugs section
- The section on library and include statements

)

LET'S DO A QUICK EXAMPLE WITH DEBUGGING

Let's also check out the power of lldb, looking at the assembly associated with the puts functions.

clang –g puts.c -o puts.out

-g : let's us do line level debugging.

TYPES IN C

type	size (bytes)
char	1
short	2
int	4
long	8
float	4
double	8

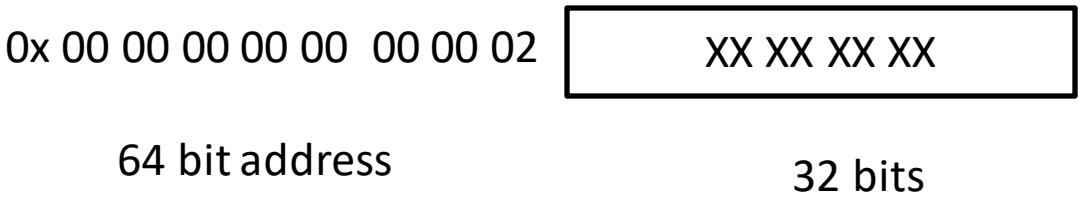
```
int x = 3;
int number_of_bytes = sizeof(x);

char letter = ‘A’;
int number_of_bytes = sizeof(letter);
```

PRINTF

Specifier	Argument	Type Example(s)
%s	char *	Hello, World!
%p	any pointer	0x4005d4
%d	int/short/char	42
%u	unsigned int/short/char	42
%x	unsigned int/short/char	2a
%ld	long	42
%f	double/float	42.000000
%e	double/float	4.200000e-19
%%	(no argument)	%

THIS DECLARES A VARIABLE

int variable;



WHAT GETS PRINTED?

```
  GNU nano 6.3    example.c     Modified
#include <stdio.h>

int main(){
     int variable;
     printf("value: %d\n", variable);
}
```

```
dgg6b@portal06:~$ clang -O3 example.c
dgg6b@portal06:~$ ./a.out
```

Is it the same every time we run the program?
What if we didn't optimize the program?

WHAT GETS PRINTED?

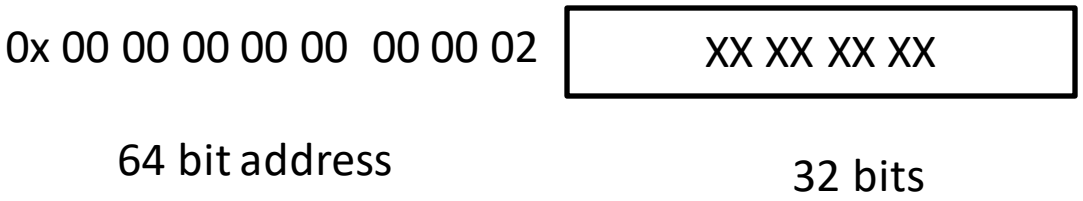
```
  GNU nano 6.3    example.c     Modified
#include <stdio.h>

int main(){
      int variable;
      printf("value: %d\n", variable);
}
```

```
dgg6b@portal06:~$ clang -O3 example.c
dgg6b@portal06:~$
```

Try not to use uninitialized variables

THIS DECLARES A VARIABLE

int variable;



WHAT IF WE RUN IT WITHOUT OPTIMIZATIONS?

Quick Demo?

Do we always want to optimize?

SCANF AND THE STACK

GNU nano 6.3 scanf.c

```c
#include <stdio.h>

int main(){
        int number;
        scanf("%d", &number);
        return 0;
}
```

Draw the stack

GNU nano 6.3 scanf.s

```
        .text
        .file   "scanf.c"
        .globl  main                            # -- Begin fun
        .p2align        4, 0x90
        .type   main,@function
main:                                           # @main
        .cfi_startproc
# %bb.0:
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset %rbp, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register %rbp
        subq    $16, %rsp
        movl    $0, -4(%rbp)
        movabsq $.L.str, %rdi
        leaq    -8(%rbp), %rsi
        movb    $0, %al
        callq   __isoc99_scanf
        xorl    %eax, %eax
        addq    $16, %rsp
        popq    %rbp
        .cfi_def_cfa %rsp, 8
        retq
.Lfunc_end0:
        .size   main, .Lfunc_end0-main
        .cfi_endproc
                                                # -- End function
        .type   .L.str,@object                  # @.str
        .section        .rodata.str1.1,"aMS",@progbits,1
.L.str:
        .asciz  "%d"
        .size   .L.str, 3
```

SCANF WRITES THE INPUT THE ADDRESS

```
 GNU nano 6.3                    scanf.s
        .text
        .file   "scanf.c"
        .globl  main                            # -- Begin function
        .p2align        4, 0x90
        .type   main,@function
main:                                   # @main
        .cfi_startproc
# %bb.0:
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset %rbp, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register %rbp
        subq    $16, %rsp
        movl    $0, -4(%rbp)
        movabsq $.L.str, %rdi
        leaq    -8(%rbp), %rsi
        movb    $0, %al
        callq   __isoc99_scanf
        xorl    %eax, %eax
        addq    $16, %rsp
        popq    %rbp
        .cfi_def_cfa %rsp, 8
        retq
.Lfunc_end0:
        .size   main, .Lfunc_end0-main
        .cfi_endproc
                                        # -- End function
```

```
dgg6b@portal03:~$ clang -g scanf.c -o scanf.out
dgg6b@portal03:~$ lldb scanf.out
(lldb) target create "scanf.out"
Current executable set to '/u/dgg6b/scanf.out' (x86_64).
(lldb) b 6
Breakpoint 1: where = scanf.out`main + 36 at scanf.c:6:2, address =
 0x0000000000401154
(lldb) run
Process 4072518 launched: '/u/dgg6b/scanf.out' (x86_64)
3405689018
Process 4072518 stopped
* thread #1, name = 'scanf.out', stop reason = breakpoint 1.1
    frame #0: 0x0000000000401154 scanf.out`main at scanf.c:6:2
   3    int main(){
   4            int number;
   5            scanf("%d", &number);
-> 6            return 0;
   7    }
(lldb) mem read $rbp-8 -fX
0x7fffffffe428: 0xBA
0x7fffffffe429: 0xB0
0x7fffffffe42a: 0xFE
0x7fffffffe42b: 0xCA
0x7fffffffe42c: 0x00
0x7fffffffe42d: 0x00
0x7fffffffe42e: 0x00
0x7fffffffe42f: 0x00
(lldb)
```

Draw the stack

GNU nano 6.3 scanf.c

```c
#include <stdio.h>

int main(){
        int number;
        scanf("%d", &number);
        return 0;
}
```

GNU nano 6.3 scanf.s

```
        .text
        .file   "scanf.c"
        .globl  main                                    # -- Begin fu
        .p2align        4, 0x90
        .type   main,@function
main:                                                   # @main
        .cfi_startproc
# %bb.0:
        pushq   %rax
        .cfi_def_cfa_offset 16
        leaq    4(%rsp), %rsi
        movl    $.L.str, %edi
        xorl    %eax, %eax
        callq   __isoc99_scanf
        xorl    %eax, %eax
        popq    %rcx
        .cfi_def_cfa_offset 8
        retq
.Lfunc_end0:
        .size   main, .Lfunc_end0-main
        .cfi_endproc
                                                        # -- End function
        .type   .L.str,@object                          # @.str
        .section        .rodata.str1.1,"aMS",@progbits,1
.L.str:
        .asciz  "%d"
        .size   .L.str, 3

        .ident  "clang version 14.0.6 (https://github.com/llv
        .section        ".note.GNU-stack","",@progbits
        .addrsig
```

THIS DECLARES A POINTER

```
int *pointer;
```

Be careful with uninitialized pointers: if referenced to without setting, it will lead to a memory error

THIS INITIALIZES A VARIABLE

int variable = 3;