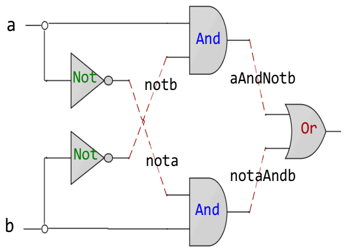
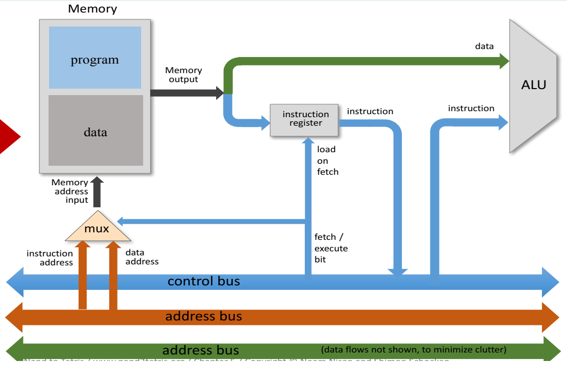
Computer Organization & Assembly Language Programming



```
CHIP Xor {
   IN a, b;
   OUT out;
   PARTS:
   Not(in=a, out=nota);
   Not(in=b, out=notb);
   And(a=nota, b=b, out=w1);
   And(a=a, b=notb, out=w2);
   Or(a=w1, b=w2, out=out);
}
```



```
@R1
D=M
@temp
M=D
```

```
0000000000000001
1111110000010000
0000000000010000
1110001100001000
```

Lecture # 45

Mixing C with x86-64 Assembly

```
#include<stdio.h>
#include<stdlib.h>
int main(){
  printf("Learning is fun with Arif\n");
  exit(0);
}
```

```
global main
SECTION .data
   msg: db "Learning is fun with Arif", 0Ah, 0h
   len_msg: equ $ - msg
SECTION .text
   main:
      mov rax,1
      mov rdi,1
      mov rsi,msg
      mov rdx,len_msg
      syscall
      mov rax,60
      mov rdi,0
      syscall
```

```
0:  b8 01 00 00 00
5:  bf 01 00 00 00
a:  48 be 00 00 00 00 00
11: 00 00 00
14: ba 1b 00 00 00
19: 0f 05
1b: b8 3c 00 00 00
20: bf 00 00 00 00
25: 0f 05
```

For resources visit my personal website:
https://www.arifbutt.me
and course bitbucket repository:
https://bitbucket.org/arifpucit/coal-repo

Instructor: Muhammad Arif Butt, Ph.D.

Today's Agenda

- Recap: Points to Ponder
- Calling C-Library Functions from Assembly Program
  - Demo (***callputs.nasm***)
  - Demo (***callprintf.nasm***)
  - Demo (***callgetputchar.nasm***)
- Calling Assembly Functions from C Program
  - Demo (***ex1.c*** and ***getval.nasm***)
  - Demo (***ex2.c*** and ***maxofthree.nasm***)

Points to Ponder

Points to Ponder

- Every microprocessor x86, MIPS, ARM, Sun SPARC, Motorola Power PC, and so on, has its own assembly language, and organizational structure. In this part of the course we are studying the assembly language of x86-64 microprocessor, which can be written in two formats Intel and AT&T
- NASM, YASM, GAS, FASM, MASM are different assemblers that can assemble the assembly language programs written for x86 microprocessors. Each assembler has its own way of writing the assembly program and has its own assembler directives. In this part of the course we are using the Netwide Assembler
- Different assemblers generate different object file format (as per the processor and OS) from the assembly source files like ELF32, ELF64, COFF, win32 and so on
- There can be different operating systems (Linux, Windows, OS/X), and the differences may come into play when we use operating system services using their respective system call interface. We are using System-V AMD64 ABI
- Finally using library functions from your assembly programs also make the difference since all linkers do not work the same way. We are using Linux linker (ld) in this part of the course and may use gcc as well

Calling C-Library Functions from Assembly Program

Calling C-Functions from Assembly Program

- Reasons to do so are:
  - There are tens of standard C library functions that can be used for I/O, specially while working with floating point numbers
  - There are extensive set of functions available in the math library, thus making our life easy

```
$ nasm –felf64 callputs.nasm
$ gcc --static callputs.o
$ ./a.out
Learning is fun with Arif Butt...
$ echo $?
34
```

```
; COAL Video Lecture: 45
;  callputs.nasm
SECTION .data
 msg: db "Learning is fun with Arif Butt..."
SECTION .text
  global main
  extern puts     ; int puts(const char *s)
  extern exit     ; void exit(int status)
main:
  lea rdi, msg
  call puts
  mov rdi, rax
  call exit
```

Examples: Calling C-Functions from Assembly Program

Demo

45/callputs.nasm
45/callprintf.nasm
45/getputchar.nasm

Calling Assembly Functions from C-Program

Calling Assembly Functions from C Program

Reasons to do so are:

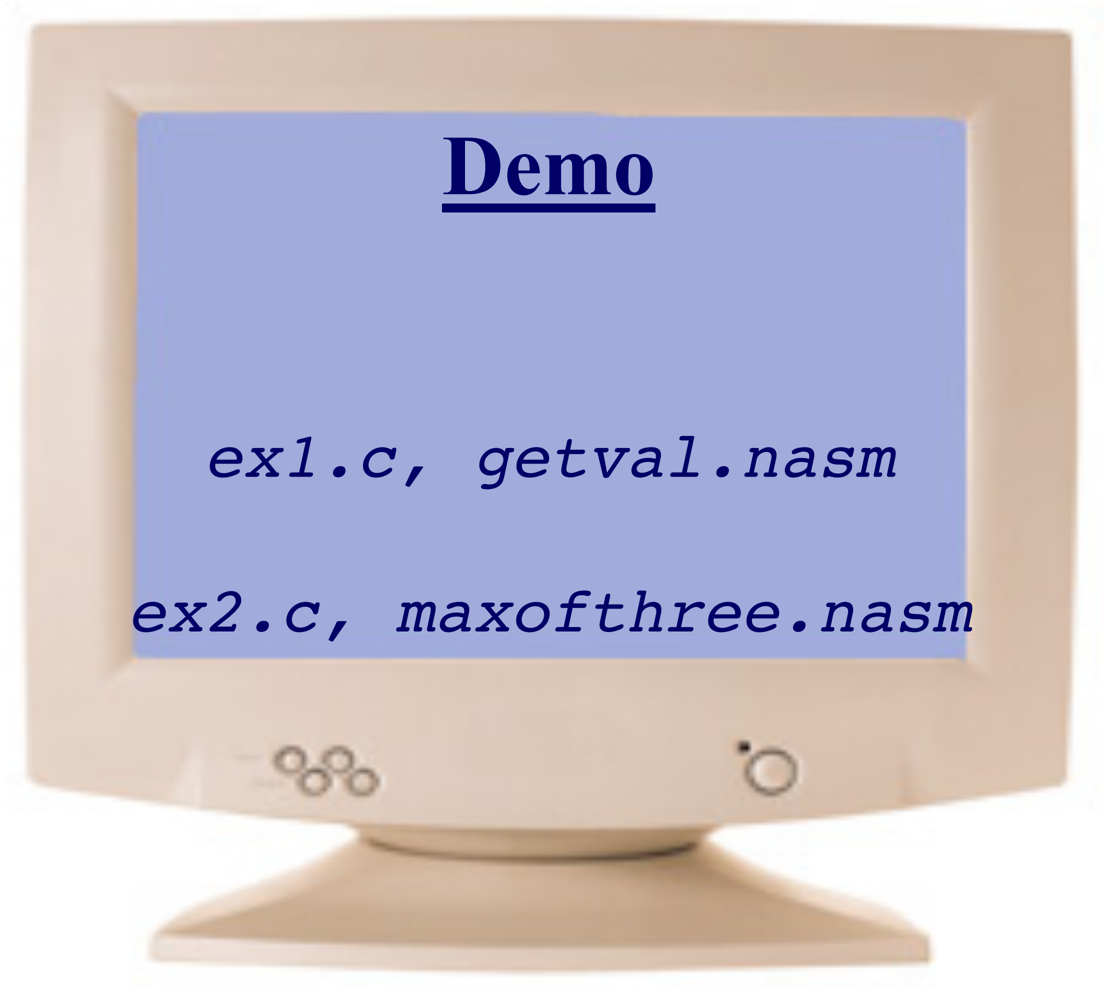
- You have assembly code already written that you wish to use
- You need to improve the speed of a particular function
- You want to manipulate SFRs or memory-mapped I/O devices

```c
// COAL Video Lecture: 45
//  ex2.c
#include <stdio.h>
#include <stdlib.h>
extern long maxofthree(long, long, long);
int main(){
   int rv = maxofthree(15, -23, 7);
   printf("max = %ld\n",rv);
   return 0;
}
```

```nasm
; COAL Video Lecture: 45
;  maxofthree.nasm
SECTION .text
  global  maxofthree
maxofthree:
  mov       rax, rdi
  cmp       rax, rsi
  cmovl     rax, rsi
  cmp       rax, rdx
  cmovl     rax, rdx
  ret
```

It is also possible to include a bit of assembly code right inside your C file, called "inline assembly". Syntax is of course compiler dependent. In gcc: `(_asm_"assembly code");`

Examples: Calling Assembly Procedures from C-Program



Demo

ex1.c, getval.nasm

ex2.c, maxofthree.nasm

Things To Do

Coming to office hours does NOT mean you are academically week!