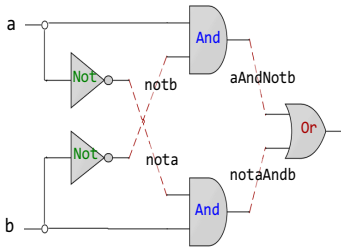
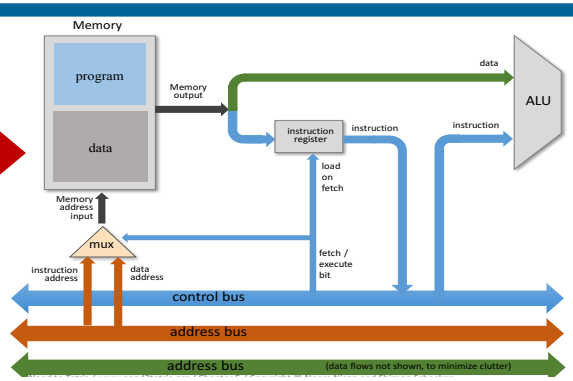
Computer Organization & Assembly Language Programming




```
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 # 41

GDB with PEDA Plugin

```
#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: GNU Debugger (**`gdb`**)
- Download and Configure PEDA plugin
- Demo (***`example.nasm`***)

Recap: GNU Debugger (gdb)

- A debugger is a program running another program (gdb, radare2, IDA, Immunity debugger, Softice, …)
- GDB is a portable debugger that can
  - handle the assembly of processors like IA-32, x86-64, arm, mips, sparc …
  - run on most popular UNIX and Microsoft Windows variants, as well as on Mac OS
  - work for many programming languages including Assembly, C/C++, Objective C, OpenCL, Go, Modula-2, Fortran, Pascal and Ada
- During this course, we have used GDB's CLI and TUI to understand what is going on inside our assembly programs while they execute and manipulating the flow of program execution
- We can use GDB for reverse engineering, cracking binaries and exploit development as well, however, gdb do not have commands for exploit development and has weak scripting support
- So to enhance the fire power of gdb for analyzing, exploiting and doing reverse engineering on executables, hackers use a gdb plug-in called **PEDA (Python Exploit Development Assistance)**

Download and Use PEDA Plugin

- Python Exploit Development Assistance (PEDA) is like an add-on/extension/plugin for GDB used extensively in exploit development, available only on Linux and supported by **`gdb 7.x`** and **`Python 2.6+`**

- **Visit:** `https://github.com/longld/peda`
- **Download and install PEDA:**

```
$ git clone https://github.com/longld/peda.git ~/peda
$ echo "source ~/peda/peda.py" >> ~/.gdbinit
```

- **Usage:**

```
$ gdb <executable>
gdb-peda$
```

Downloading PEDA

Example: Visualizing Stack using gdb-peda

```
; COAL Video Lecture: 41
;  example.nasm
SECTION .text
  global _start
  _start:
    mov rax, 255
    xor rax, rax
    push 65   ; 0x41 ('A')
    push 66   ; 0x42 ('B')
    push 67   ; 0x43 ('C')
    pop r11
    pop r12
    pop r13
;exit gracefully
    mov rax, 60
    mov rdi, 0
    syscall
```

- In a process logical address space, the stack is at the top of memory and grows from higher memory addresses to lower memory addresses in architectures like Intel, MIPS, Motorola, SPARC
- The stack pointer `rsp` always contains address of current top of stack, i..e., it points to the last inserted item
- All the push/pop on the stack are 8 Bytes wide on x86_64

Assembling & Executing x86-64 Program

Things To Do

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