



x86 Assembly Tutorial

COS 318: Fall 2019



Project 1 Schedule

- Design Review: Monday 9/23 & Tuesday 9/24
 - [Sign up](#) for 10-min slot from 3:00pm to 7:00pm
 - Complete set up and answer posted questions
- (Official) Precept: Monday 9/23 & Tuesday 9/24, 7:30pm - 8:20pm
- Due: Sunday 9/29, 11:55pm



Overview

- x86 (*technically* IA-32) Assembly Overview
 - Registers, Flags, Memory Addressing, Instructions, Stack / Calling Convention, Directives, Segments
- BIOS + GDB
- Design Review



Registers

General Purpose Registers: 8,16,32 bits

31	15	7	0
	AH	AL	AX = AH AL
	BH	BL	BX = BH BL
	CH	CL	CX = CH CL
	DH	DL	DX = DH DL
	BP		
	SI		
	DI		
	SP		

EAX

EBX

ECX

EDX

EBP

ESI

EDI

ESP

Segment Registers: 16 bits

CS
DS
SS
ES
FS
GS

Instruction Pointer (EIP): 32 bits

Flags (EFLAGS): 32 bits



Flags

- Function of flags
 - Control the behavior of CPU
 - Save the status of the last instruction
 - Details: https://en.wikipedia.org/wiki/FLAGS_register



Flags

- Important flags:
 - CF: Carry flag
 - ZF: Zero flag
 - SF: Sign flag
 - IF: Interrupt flag (sti, cli)
 - DF: Direction flag (std, cld)



AT&T (vs Intel) Syntax

- Prefix register names with % (e.g. %ax)
- Instruction format: **instr src,dest**
 - `movw %ax,%bx`
- Prefix constants (immediate values) with \$
 - `movw $1,%ax`
- Suffix instructions with size of data
 - b for byte, w for word (16 bits), l for long (32 bits)



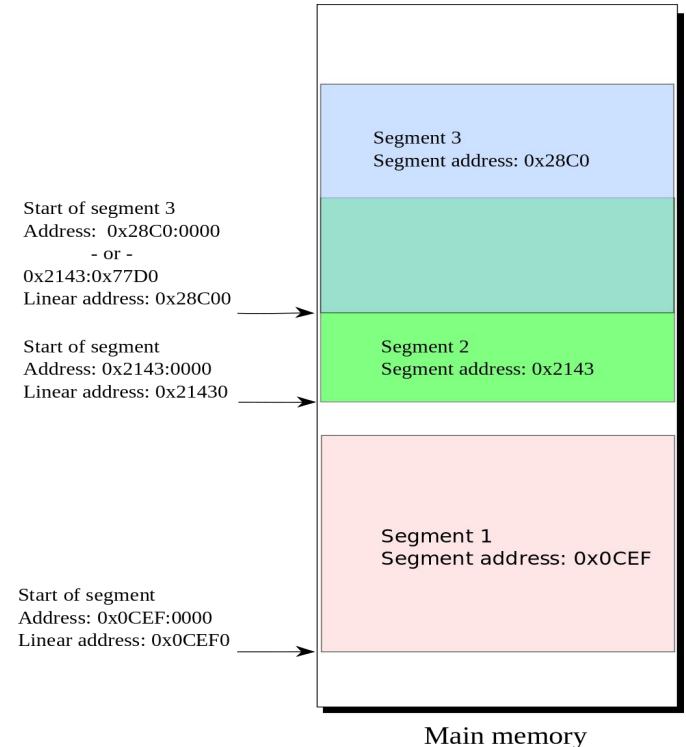
Memory Addressing (Real Mode)

- 1MB memory
 - Valid address range: 0x00000 ~ 0xFFFFF
- See full 1MB with 20-bit addresses
- 16-bit segments and 16-bit offsets



Memory Addressing (Real Mode)

- Format (AT&T syntax):
 - **segment:displacement(base,index,scale)**
- $\text{Offset} = \text{Base} + \text{Index} * \text{Scale} + \text{Displacement}$
- $\text{Address} = (\text{Segment} * 16) + \text{Offset}$
- Displacement: Constant
- Base: %bx, %bp
- Index: %si, %di
- Segment: %cs, %ds, %ss, %es, %fs, %gs





Instructions: Arithmetic & Logic

- **add/sub{l,w,b} source,dest**
- **inc/dec/neg{l,w,b} dest**
- **cmp{l,w,b} source,dest**
- **and/or/xor{l,w,b} source,dest ...**
- Restrictions
 - No more than one memory operand



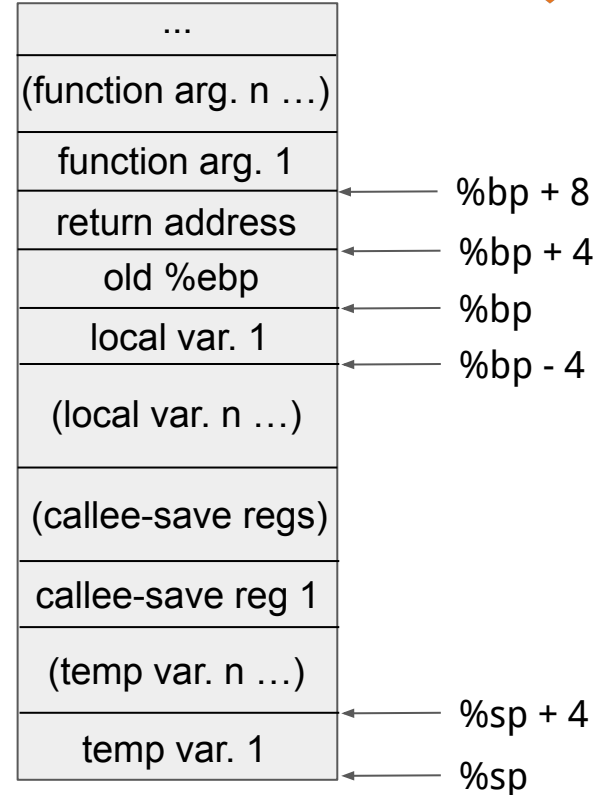
Instructions: Data Transfer

- **mov{l,w,b} source, dest**
- **xchg{l,w,b} source, dest**
- movsb/movsw
 - $\%es:(\%di) \leftarrow \%ds:(\%si)$
 - Often used with %cx to move a number of bytes
 - movw \$0x10,%cx
 - rep movsw
- Segment registers can only appear with registers



Stack Layout

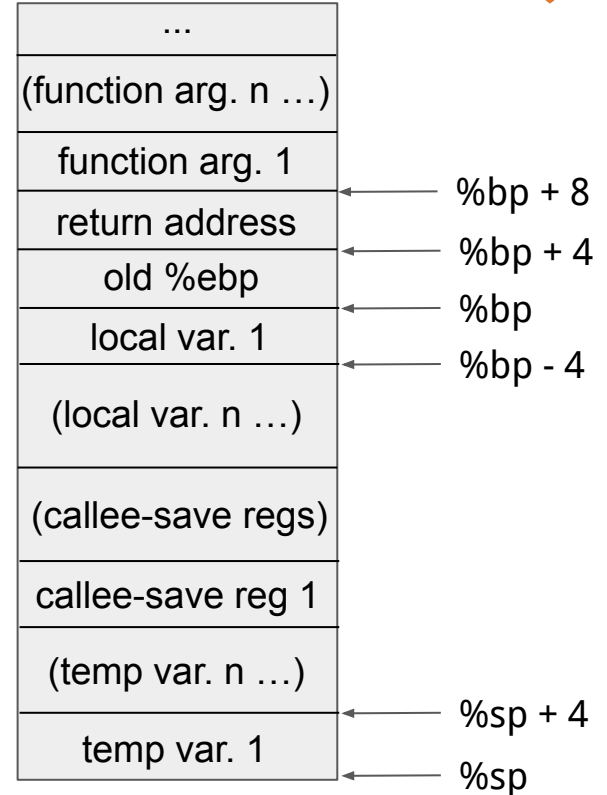
- Grows from high to low
 - Lowest address = “top” of stack
- `%sp` points to top of the stack
 - Used to reference temporary variables
- `%ebp` points to bottom of stack frame
 - Used for local vars + function args.





Calling Convention

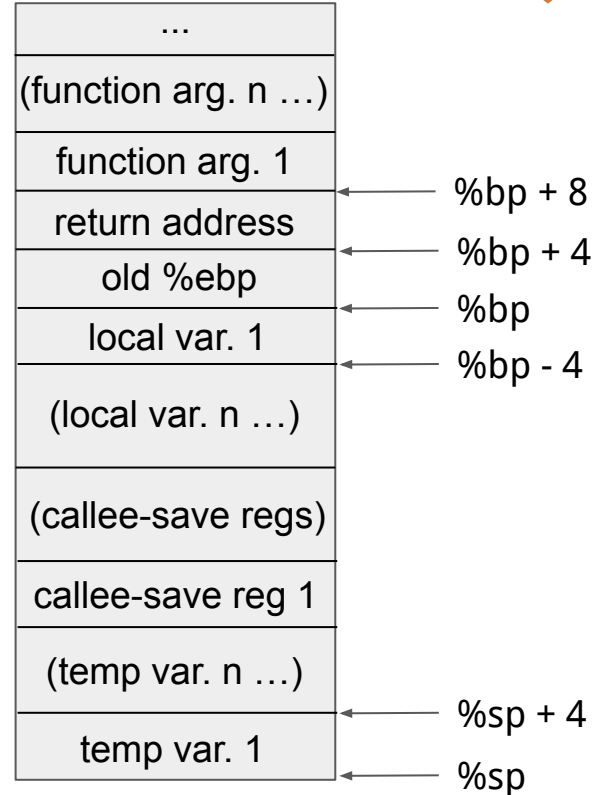
- When calling a function:
 - 1. Push caller-save regs onto stack
 - 2. Push function args onto stack
 - 3. Push return address + branch
- In subroutine:
 - 1. Push old %bp + set %bp = %sp
 - 2. Allocate space for local variables
 - 3. Push callee-save regs if necessary





Instructions: Stack Access

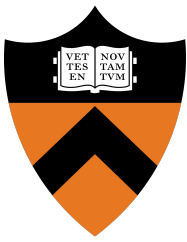
- **pushl source**
 - $\%sp \leftarrow \%sp - 4$
 - $\%ss:(\%sp) \leftarrow \text{source}$
- **popl dest**
 - $\text{dest} \leftarrow \%ss:(\%sp)$
 - $\%sp \leftarrow \%sp + 4$





Instructions: Control Flow

- **jmp label**
 - $\%eip \leftarrow \text{label}$
- **ljmp NEW_CS, offset**
 - $\%CS \leftarrow \text{NEW_CS}$
 - $\%eip \leftarrow \text{offset}$
- **call label**
 - push $\%eip$
 - $\%eip \leftarrow \text{label}$
- **ret**
 - pop $\%eip$



Instructions: Conditional Jump

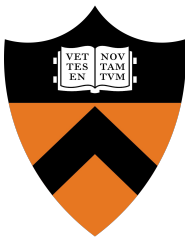
- Relies on %eflags bits
 - Most arithmetic operations change %eflags
- **j* label**
 - Jump to label if * flag is 1
- **jn* label**
 - Jump to label if * flag is 0



Assembler Directives

- Commands that speak directly to the assembler
 - Are not instructions
- Examples:
 - `.globl` - defines a list of symbols as global
 - `.equ` - defines a constant (like `#define`)
 - `.bytes`, `.word`, `.asciz` - reserve space in memory

https://docs.oracle.com/cd/E26502_01/html/E28388/eoiyg.html



Assembler Segments

- Organize memory by data properties
 - `.text` - holds executable instructions
 - `.bss` - holds zero-initialized data (e.g. `static int i;`)
 - `.data` - holds initialized data (e.g. `char c = 'a';`)
 - `.rodata` - holds read-only data
- Stack / Heap - Set up by linker / loader / programmer



BIOS Services

- Use BIOS services through int instruction
 - Must store parameters in specified registers
 - Triggers a software interrupt
- **int INT_NUM**
 - int \$0x10: Video services
 - int \$0x13: Disk services



Useful GDB Commands

- r - show register values
- sreg - show segment registers
- s - step into instruction
- n - next instruction
- c - continue
- u <start> <stop> - disassembles C code into assembly
- b - set a breakpoint
- d <n> - delete a breakpoint
- bpd / bpe <n> - disable / enable a breakpoint
- x/Nx addr - display hex dump of N words, starting at addr
- x/Ni addr - display N instructions, starting at addr



Design Review

- USBs will be given at the Design Review
- Write `print_char` and `print_string` assembly functions
- Be ready to describe:
 - How to move the kernel from disk to memory
 - How to create disk image
 - (More specific guidelines are provided on the project page)