# Comp 3350: Computer Organization & Assembly Language

# HW # 2: Theme: x86 Organization Basics

*All main questions carry equal weight.*

*(Credit awarded to only those answers with work shown)*

1. Name all eight 32-bit general-purpose registers. What is the general function of each of the registers? Which of these registers can be addressed in parts?

**EAX:** Automatically used by multiplication & division instructions; primarily used for accumulation.

**EBX:** Has no specific uses but is often set to a commonly used value to speed up calculations in a function. Otherwise, it can be used in indexed addressing.

**ECX:** Used by the CPU as a loop counter.

**EDX:** Occasionally used as a function parameter. Generally used for storing short-term variables within a function. Has a part in multiplication/division operations involving large values, and can also be used in I/O operations.

**EBP:** References function parameters & local variables on the stack for high-level languages as an extended frame pointer.

**ESP:** Rarely used for ordinary arithmetic or data transfer, although mostly used to address data on the stack.

**ESI:** Used by high-speed memory transfer instructions and as an index register.

**EDI:** Used by high-speed memory transfer instructions and as an index register.

All the general registers can be addressed in parts. EAX, EBC, ECX, & EDX can all be accessed via 16-bit registers called AX, BX, CX, & DX or 8-bit (High) or 8-bit (Low) registers, such as AH, BH, CH, DH, AL, BL, CL, & DL. The remaining 4 general-purpose registers can instead be accessed by 16-bit names, named SI, DI, BP, & SP.

1. What do the Sign Flag, Zero Flag, Auxiliary Carry Flag, and Parity Flag indicate when set?

The SF (when set) indicates that an arithmetic / logical operation generated a negative result, the ZF (when set) indicates that an arithmetic / logical operation generated a result of zero, the AC (when set) indicates an arithmetic operation caused a carry from bit 3 to bit 4 in an 8-bit operand, & the PF (when set) indicates the least-significant byte in the result contains an even number of 1 bits (this is mostly used for error checking).

1. What do the Overflow Flag and Carry Flag indicate when set? How do they differ?

The OF (when set) indicates the result of a signed arithmetic operation is too large or too small to fit into the destination. The CF (when set) indicates the result of an unsigned arithmetic operation is too large to fit into the destination. The difference between the two is that the first relies on an indication of a signed arithmetic operation and whether it is too small or too large to fit into a destination, while the second relies on an indication of an unsigned arithmetic operation and only cares if it is too large to fit into the destination.

1. Detail the process by which instructions and data are read from memory.

Instructions & data are held in the memory storage unit while a computer program is running. The storage unit receives requests for data from the CPU, transfers data from RAM to the CPU, and transfers data from the CPU into memory via the data bus. All processing of data takes place within the CPU, so programs residing in memory must be copied into the CPU before they can execute. Individual program instructions can be copied into the CPU one at a time, or groups of instructions can be copied together. The address bus is then responsible for holding the addresses of instructions and data when the currently executing instruction transfers data between the CPU & memory.

1. What do you understand by Cache memory and what are its benefits?

Cache memory comes in two types for the x86 family: level-1 cache & level-2 cache. The level-1 cache is otherwise known as the primary cache and is stored right on the CPU. The level-2 cache is otherwise known as the secondary cache and is slower as a result of being attached to the CPU by a high-speed data bus. We prefer cache memory over RAM because cache memory is constructed from static RAM. The benefit to this is that it does not have to be constantly refreshed in order to keep its contents and is therefore incredibly fast, although it is more expensive than dynamic RAM, which must be refreshed constantly and is therefore slower but costs less as a tradeoff.

1. What do you understand by Real-address mode, Protected mode, Multi-segment model? *Discuss in detail.*

**Real-address mode:** Only 1MB of memory can be addressed, from hexadecimal 00000 to FFFFF. The processor can run only one program at a time, but it can momentarily interrupt that program to process requests (called interrupts) from peripherals. Application programs are permitted to access any memory location, including addresses that are linked directly to system hardware. The MS-DOS operating system runs in real-address mode, and Windows 95 & 98 can be booted into this mode.

**Protected mode:** The processor can run multiple programs at the same time. It assigns each process (running program) a total of 4GB of memory. Each program can be assigned its own reserved memory area, and programs are prevented from accidentally accessing each other’s code and data.

**Multi-segment model:** Each program has a Local Descriptor Table (LDT). This table holds a descriptor for each segment used by the program. This provides the full capabilities of the segmentation mechanism to the program.

1. In a 32-bit computer what are the maximum memory amounts that can be addressed in (a) real-addressed mode (b) protected mode? What is the linear address corresponding to the following segment-offset: 04B2:2033?
2. In real-address mode programs, a range of 1MB can only be addressed.
3. In protected mode programs, each program can have its own 1MB memory area, but a task or program can address a linear address space of up to 4GB’s.

Calculating a linear address for a segment-offset requires tacking a zero onto the end of the first part, and then adding the second part to it. The resulting hexadecimal number will be the linear address. For this specific segment-offset, we just add 04B20 and 2033 together to get 24E5 as the linear address.

1. Let us say your computer is running at 3 GHz. You come to know that the Add instruction takes 6 clock periods in your computer. Express the time taken by the Add instruction in nanoseconds.

Since the computer runs at 3 GHz, this means the clock oscillates 3 billion times per second. If the Add instruction takes 6 clock periods, and each clock period is 1/3,000,000,000th of a second, this means the Add instruction takes 6/3,000,000,000th of a second, which is equal to 2 nanoseconds.