Lecture 5: Computer Organization
Instruction Execution

- Computer Organization
- Registers
- Addressing
- Buses
- Fetch-Execute Cycle

Components

- Control Unit – fetches instructions, decodes instructions, causes instructions to be carried out.
- Arithmetic logical unit (ALU) – performs arithmetic operations (addition, etc.) on data.
- Registers – high speed memory cells (don’t need to go through the bus to access). They vary in number and purpose on different machines.
- Buses – communication pathways connecting different devices/components.

Registers

- 8, 16, or 32 bit high-speed storage locations inside the CPU
- They can be accessed at a much higher speed than conventional memory.
- When optimizing for speed, use registers.
- Four types: general purpose, segment, index, status, and control

General Purpose Registers

- Data registers, also known as general purpose registers: AX, BX, CX, DX
- Used for arithmetic operations and data movement
- Can be addressed as 16 bit or 8 bit values. For AX, upper 8 bits are AH, lower 8 bits are AL.
- Remember: when when a 16 bit register is modified, so is the corresponding 8 bit registers!
Special Attributes of GP Registers

- AX – accumulator – fastest for arithmetic operations. Some math instructions only use AX.
- BX – base – this register can hold an address of a procedure or variable. BX can also perform arithmetic and data movement.
- CX – counter – this register acts as a counter for repeating or looping instructions
- DX – data – this register has a special role in multiply and divide operations. In multiplication it holds the high 16 bits of the product. In division it holds the remainder.

Segment Registers

- Segment registers are used as base locations for program instructions, data, and the stack.
- All references to memory involve a segment register as the base location.

Segment Registers, cont.

- CS – code segment – this register holds the base location of all instructions in a program
- DS – data segment – this is the default base location for variables. It is used by the CPU to calculate the variable location.
- SS – stack segment – this register contains the base location of the stack.
- ES – extra segment – this is an additional base location for memory variables.

Index Registers

- Index registers contain the offsets of data and instructions.
- Offset refers to the distance of a variable, label, or instruction from its base segment.
- Index registers are used when processing strings, arrays, and other data structures.

Index Registers, cont.

- BP – base pointer – this register contains an offset from the SS register and is often used by subroutines to find the variables passed to it on the stack.
- SP – stack pointer – this register contains the offset from the top of the stack. The complete top of stack address is calculated using the SP and SS registers.
- SI – source index – used to point to data in memory. Named because this is the index register commonly used as the source in string operations (for example)
- DI - destination index – index register commonly used as the destination in string operations.
Status and Control Registers

- IP – instruction pointer – always contains the offset of the next instruction. The IP and CS registers combine to form the complete address.
- Flags – a special register with individual bit positions that give the status of the CPU (control flags) or results of arithmetic operations (status flags).

Control Flags

- Direction flag (DF) – affects block data transfer instructions. Indicates if you are moving up (0) or down (1) in memory.
- Interrupt flag (IF) – indicates if system interrupts can occur. Interrupts will be disabled if a critical operation is being performed that cannot be interrupted.
- Trap flag (TF) – determines if the CPU is halted after each instruction. If it’s set, a debugger will allow single stepping through the program.

Status Flags

- These indicate the status of arithmetic and logical operations.
- Carry flag (CF) – set if the result of an unsigned operation is too big to fit into the destination. 1 = carry, 0 = no carry.
- Overflow flag (OF) – set if the result of a signed operation is too wide to fit into the destination. 1 = overflow, 0 = no overflow.
- Sign flag (SF) – set when the result of an operation is negative. 1 = negative, 0 = positive

Status Flags, cont.

- Zero flag (ZF) – set when the result of an arithmetic operation is zero. Used by branch and loop instructions when comparing values. 1 = zero, 0 = not zero.
- Auxiliary Carry – set when an operation causes a carry from bit 3 to bit 4 or a borrow from bit 4 to bit 3. 1 = carry, 0 = no carry.
- Parity – indicates if the result of an operation has an even or odd number of bits. Used to verify memory integrity or correct transmission of data.

CPU: Instruction Cycle

- Three basic operations:
  - Fetch: fetches the instruction, copying it from memory into the instruction queue, and increments the PC (program counter).
  - Decode: control unit determines the type of the instruction. Retrieves any needed operands and sends them to the ALU. Also sends the ALU signals to indicate the type of operation.
  - Execute: ALU executes the instruction, returns output to the destination, and update the status flags.
- Each step takes one click of the system clock: one clock cycle.

Addressing

- Address: a number referring to an 8-bit memory location.
- Logical addresses go from 0 to the highest location.
- How these are translated into physical addresses varies.
- For Intel:
  - 32-bit segment-offset address: combination of base location (segment) and offset to represent a logical location
  - 20-bit absolute address, which refers to a physical memory location.
Addressing, cont.

- Problem: how to address 1,048,576 bytes of memory with a 16-bit wide address register (where the max is 65,535)
- Solution: combine segment and offset values to obtain the absolute address
- Example: 08F1:0000
  1) convert segment to absolute by adding 4 zero bits: 08F10
  2) add the offset: 0100 (hex)

  08 100 0100 08F10
  + 0 010 0100 0100
  ---------------
  09010100 09010
  --
  segment value w/extra 40 bits
  --
  obtain the absolute address
  (effective address)

Why Segment-Offset?

- You can load the program at any segment address and individual variable addresses to not need to be recalculated.
- Why? Variable locations are 16-bit offsets from the program’s data area.
- This is known as being segment relocatable
- Programs can access large data structures by modifying the segment portion of the data’s address to point to new blocks of memory.

Data Segment

addressable memory on 8086

Example:
- DS = 0100h
- data seg. start?
- data seg. end?

If BX contains the offset:
- BX = 005Ah
- EA = ?

Segment Register Combinations

- Code Segment – the CS register and IP (instruction pointer) are used to point to the next instruction.
- Stack – the SS register is used with the SP (stack pointer) or BP (base pointer)
- Data Segment – DS with BX, SI, or DI
- Extended Segment – BX, SI, or DI

More on Effective Addresses

- There’s more than one way to get the same effective address!
- Example:
  - CS = 147Bh
  - IP = 131Ah
  - EA = 147B0 + 131A = 15ACAh

  or
  - CS = 154Ch
  - IP = 000Ah
  - EA = 15AC + 000A = 15ACAh

- If CS = 147B, what range of effective addresses can be referenced without changing the value in CS?
Buses

• Communication pathway connecting devices.
• Set of parallel wires (lines) connecting components.
• Serial – over time, a sequence of binary digits can be passed along a single line.
• Parallel – one line for each bit (8-bit unit of data can be transmitted over 8 bus lines). The number of bits is called the bus “width”

System Buses

• Connects major components (CPU, memory, I/O)
• Data bus – used for sending data between CPU/memory. Usually as many lines as bits in a word.
• Address bus – when CPU wants to read/write memory, needs to tell memory which word it wants to access. Usually same number of lines as address size (20 bits for us)
• Control bus – tells memory whether CPU wants to read or write. Transmits both commands (operation to be performed) and timing information.

Control Lines

• Example:
  – memory read
  – memory write
  – I/O read
  – I/O write
  – Bus request
  – Bus grant
  – Transfer ACK
  – Interrupt request
  – Interrupt ACK
  – Clock (synchronous operations)
  – Reset (init. all modules)

Operation of Bus

• One module wants to send data to another:
  1. obtain use of bus
  2. transfer data via the bus
• One module wishes to request data from another
  1. obtain use of the bus
  2. transfer request to the other module over appropriate address and control lines
  3. wait for the other module to send data
• Shared transmission medium!
  – Signal transmitted by one device is available to all
  – Only one device at a time can successfully transmit or signals will be garbled.

Example

• Fetching an instruction:
  – CPU puts the address of the instruction on the address pins
  – Asserts “memory read”
  – Memory replies by putting instruction on data pins
  – Asserts a signal that it’s done (READY)
  – When CPU sees the signal, it accepts the word and carries out the instruction.

Pins on the 8086
Pin Explanations

- S0-S2 – Bus Control
  - INTA, I/O read, I/O write, HALT, code access, memory read, memory write, latches on bus
- S3-S6, Q0-Q1 – Status of internal state (not normally used)
- RD – read cycle taking place
- LOCK – leave bus alone (during critical CPU instructions that require multiple bus cycles)
- READY – when CPU asks for a byte from memory, memory is expected to deliver within 4 bus cycles. If memory is too slow, it negates Ready and keeps it negated until the byte is put on the bus.
- RQGrant – bus arbitration (for example: between CPU and co-processor)
- A0-A19 – address bus
- D0-D15 – data bus

Example

- Example: CPU executes instruction to fetch contents of word D0126 from memory
- CPU puts D0126 on address bus
  1101 0000 0001 0010 0110
- At the same time, asserts memory read on control bus (using pins 26-28 on diagram)
- Memory sends back contents of that word on the data bus. If contents of D0126 are 0003, then memory sends:
  0000 0000 0000 0011

Inside the CPU

CPU with visible and invisible registers
IP (PC) is invisible on some machines

Fetch/Execute Cycle (Instruction Execution)

- Instruction cycle:
  - fetch (get next instruction from memory to CPU)
  - execute (interpret opcode and perform indicated operation) (interrupt if enabled and one has occurred, save state and process interrupt)
- Fetch:
  - copy contents of 16 * contents of CS + Contents of (IP) into MAR
  - assert memory read in control bus
  - wait for memory to send back this 1 byte (or word) along data bus to MBR
  - IP is incremented (to point to the next byte)

Fetch/Execute (cont.)

- Execute:
  - CPU copies MBR into IR (so MBR can be used for future memory accesses)
  - CPU inspects opcode in IR and decodes. If other instruction bytes are needed, they are fetched as above (with IP incremented each time)
  - any operands required from memory are fetched now (put address on MAR, wait for receipt in MBR)
  - INSTRUCTION EXECUTES
  - If instruction stores results back in memory, result in MBR, put address in MAR, assert memory write.