CSCI 4717/5717
Computer Architecture

Topic: CPU Registers

Reading: Stallings, Sections 12.1 and 12.2

CPU Internal Design Issues

- CPU design and operating system design are closely linked
- Compiler also has heavy dependence on CPU design

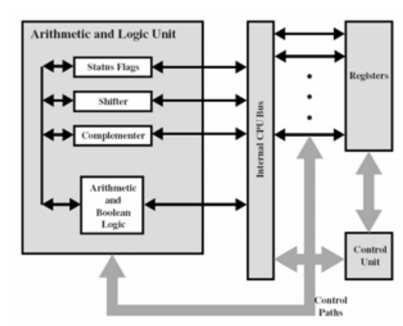
CPU Internal Design Issues

From our discussion of the architecture of the computer, we've put some requirements on the CPU:

- – Fetch instructions from memory
- – Interpret instructions to determine action that is required
- – Fetch data that may be required for execution (could come from memory or I/O)
- – Process data with arithmetic, logic, or some movement of data
- – Write data to memory or I/O

CPU Internal Structure

Design decisions here affect instruction set design



CPU Internal Structure (continued)

- Arithmetic Logic Unit
  - – Status flags
  - – Shifter
  - – Complementer
  - – Arithmetic logic
  - – Boolean logic
- Internal CPU bus to pass data back and forth between items of CPU

CPU Internal Structure (continued)

Registers

- CPU must have some working space (temporary storage) to remember things
  - – Data
  - – location of last instruction or next instruction
  - – instruction as it's working with it
- Number and function vary between processor designs
- One of the major design decisions
- Absolute top level of memory hierarchy

CPU Internal Structure (continued)

Two types of registers:

- User-visible registers -- allow for operations with minimal interaction with main memory (programmer takes place of cache controller)
- Control and Status Registers -- with correct privileges, can be set by programmer. Lesser privileges are required to read them.

CPU Internal Structure (continued)

- Control unit -- managing operation of all CPU items
- Internal CPU bus to pass data back and forth between items of CPU

User Visible Registers

- Accessed through machine/assembly language instructions
  - General Purpose
  - Data
  - Address
  - Condition Codes
- Represent complete user-oriented view of processor -- therefore, storing and later restoring effectively resets processor back to stored state

General Purpose Registers

- May be true general purpose -- can contain the operand for any opcode
- May be restricted -- floating point only, integer only, address only
- May be used for data or addressing -- some may do either address or data, in some cases there may be a clear distinction between data and address registers
- Accumulator → Data
- Addressing
  - Segment
  - Index -- may be autoindexed
  - Stack

Register Design Issues

The range of design decisions goes from…

- Make all registers general purpose
  - Increase flexibility and programmer options
  - Increase instruction size & complexity
- Make them specialized
  - Smaller more specialized (faster) instructions
  - Less flexibility

Register Design Issues (continued)

How many general purpose registers?

- Number affects instruction set design => more registers means more operand identifier bits
- Between 8 – 32
- Fewer
- Remember that the registers act as a very small cache
- The fewer GP registers, the more memory references
- More does not necessarily reduce memory references and takes up processor real estate RISC needs are different and will be discussed later

Register Design Issues (continued)

How big do we make the registers?

- Address -- large enough to hold full address
- Data -- large enough to hold full word
- Often possible to combine two data registers -- e.g. AH + AL = AX
- Example: Do we link the design of registers to a standard, e.g., C programming
  - double int a;
  - long int a;

Condition Code Registers (flags)

- Sets of individual bits each with a unique purpose (e.g. result of last operation was zero)
- Opcodes can read values to determine outcomes (e.g., conditional jumps)
- Automatically set as a result of some operations
- Some processors allow user to set or clear them explicitly
- Collected into group and referred to as a single register (CCR)

Control & Status Registers

Types of control & status registers

- Registers for movement of data between CPU and memory
  - Program Counter (PC)
  - Instruction Register (IR)
  - Memory Address Register (MAR)
  - Memory Buffer Register (MBR)
- Optional buffers used to exchange data between ALU, MBR, and user-visible registers
- Program Status Word (PSW)
- Address pointers used for control
- Built-in processor I/O control & status registers

Control & Status Registers (continued)

- Program Counter (PC)
  - Automatically incremented to next instruction as part of operation of current instruction
  - Can also be changed as result of jump instruction
- Instruction Register (IR)
  - Most recently fetched instructions
  - Where instruction decoder examines opcode to figure out what to do next

Control & Status Registers (continued)

- Memory Address Register (MAR)
  - Memory address of current memory location to fetch
  - Could be instruction or data
- Memory Buffer Register (MBR)
  - Last word read from memory (instruction or data)
  - Word to be stored to memory

Control & Status Registers (continued)

Program Status Word (PSW) – May be exactly the same thing as user-visible condition code register

- A set of bits which include condition codes
  - Sign of last result
  - Zero
  - Carry
  - Equal
  - Overflow
  - Interrupt enable/disable
  - Supervisor
    - Examples: Intel ring zero, kernel mode
    - Allows privileged instructions to execute
    - Used by operating system
    - Not available to user programs

Control & Status Registers (continued)

- Address pointers used for control
  - Interrupt vectors
  - System stack pointer
  - Page table pointer for hardware supported virtual memory
  - Chip select controls
- On processor I/O
  - Status and control to operate the I/O
  - E.g., serial ports -- bps rate, interrupt enables, buffer registers, etc.