Microprocessors

General Features

To be Examined For Each Chip

Jan 24th, 2002

Memory Structure

- Addressable units
  - For example byte vs word addressable
    - Most machines are 8-bit byte addressable
- Memory size
- Form of Addresses
  - Usually a simple linear address 0 .. 2**N-1
  - But not always

Data Formats

- We are interested in hardware support
- Integer formats
- Floating formats
- Character Formats
- Pointer/Address formats
- Other types

Integer Formats

- Sizes supported (1,2,4,8 byte)
- Unsigned vs Signed
  - Signed format (usually twos complement)
- Little-endian vs Big-endian
  - Little-endian has least significant byte at lowest address
  - Big-endian has most significant byte at lowest address.
  - May have dynamic/static switching

More on Endianness: An Example

- Suppose we have the integer 258 in decimal
- Hex value (16 bits) is 0102h
- Suppose machine is byte addressable
- Value is stored at addresses 1000h and 1001h
- Little endian:
  - 1000h: 02h
  - 1001h: 01h
- Big endian:
  - 1000h: 01h
  - 1001h: 02h

Floating Formats

- Complex possibilities
- Most machines use “IEEE” formats
  - IEEE = IEEE standard 754/854
  - Provides 32- and 64-bit standard formats
  - Also implementation dependent larger format
    - On Intel, this is 80-bits
- To be addressed separately

Character Formats

- Common character codes
  - ASCII (7-bit)
  - ISO Latin-1 (8 bit)
  - PC codes (several possibilities)
  - Unicode (16-bit)
- Machines usually do not have any specific hardware that cares what code you use

Pointer/Address Formats

- Used to hold address of memory location
- Usually but not always simple unsigned integer, using same operations
- But on some machines, segmented forms are used (to be examined later)

Other Formats

- Remember we are talking hardware here
- Packed decimal
  - Two decimal digits stored in one byte
    - For example 93 stored as hexadecimal 93
    - Used by COBOL programs
- Fractional binary
  - For example with binary point at left
  - 80000000h = 0.5 (unsigned)
- Special graphics/multimedia formats
- Vector/array processing

Data Alignment

- Consider address of a four byte integer.
- Said to be aligned if the address is a multiple of four bytes
- What happens if data is misaligned
  - Works, but slower. How much slower?
  - Fatal error, not recoverable
  - Error, but software recoverable (slow!)

General Register Structure

- How many registers?
- How many bits
- What data can be stored?
- Special purpose vs General purpose
  - Special purpose registers used only by certain instructions (e.g. ECX for loops on ia32)
  - General purpose registers fully interchangable

Specialized Registers

- Flag registers
- System registers (e.g. state of floating-point unit)
- Debug and control registers
- Special registers for handling interrupts
- Special registers for special instructions

Instruction Set

- What set of operations are available
- What memory reference instructions
  - Load/Store only vs more extensive
- What operations between registers
- How are flags set etc

Addressing Modes

- Direct addressing (address in instruction)
  - No room in RISC design for 32 bit address in a 32 bit instruction, so often not available.
- Indirect through register (simple indexing)
  - Available on all machines. Quite general since all of the instruction set can be used to compute the needed address.
- Other addressing modes

Other Addressing Modes

- Index + Offset (offset in the instruction)
  - Common, offset is small (8-16 bits)
  - Used to reference locations on stack frame
  - Used to reference fields in structure
- Double indexing (two regs added)
- Double indexing + offset
- Scaling (register multipled by 2,4,8 …)
- Fancier indirect modes

Instruction Formats

- How many different instruction formats
- Fixed vs Variable size instructions
- Uniform vs non-uniform formats
- Size of instructions

Instruction Level Parallelism

- Can instructions execute in parallel
- If so, is this
  - Invisible to programmer
    - Instructions executed as though in strict sequence
  - Visible to programmer
    - Programmer must be aware of parallelism
- Specific special cases
  - For example, branch delay slots

Branch Delay Slots

- On some machines, instructions are basically executed in sequence, except for jumps, where we have:
  - mov ...
mov …
jmp …
mov …
  - Third mov instruction is executed BEFORE the jump instruction.

Traps and Interrupts

- Terminology varies
- We will use the terms this way
  - Traps are the immediate synchronous result of a particular instruction, for example, a trap caused by a division by zero
  - Interrupts happen asynchronously because of external events, e.g. an I/O interrupt

Traps

- What instructions cause traps
- Are traps strictly synchronous?
- Or is this a place where pipelines become potentially visible
- How does machine handle trap

Interrupts

- How are external interrupts handled
- Multiple levels of interrupts
- Interrupt priorities
- Mechanisms for handling interrupts

Modes of Operation

- Kernel/supervisor/system mode vs application/program mode
- Or more fancy schemes (e.g. four levels in ia32 architectures), rings of protection.
- What instructions are disallowed when?
- How does operation switch from one mode to another?

Handling of I/O

- Input/Output Instructions
- Input/Output Channels
- Interaction with processing modes
- Interrupt handling

Memory Handling

- All machines we look at have virtual addressing capabilities.
- This is a system for mapping from virtual addresses to physical addresses
- Handled by hardware/software?
- What algorithms are used?
- Interaction with system modes

Caching Issues

- Mostly a matter of implementation rather than architecture.
- But caching may be visible to software
- For example, instructions that bypass the cache.

Parallel Processing Issues

- Consider building machine with more than one processor
- What help from hardware?
  - Synchronization
  - Cache coherency
  - Shared vs separate memory
  - Message processing
- How is shared memory handled?

MP - Synchronization

- How do processors communicate
  - Shared memory
  - Special locking instructions
  - Message passing

MP – Cache Coherency

- Suppose multiple processors share a common memory.
- Each processor has a cache
- What happens if one processor changes memory
  - Other processor may have old data in the cache (like what happens with internet browsers sometimes)

MP: Shared vs Separate Memory

- Many processors may share same memory
- If they do, how are conflicts resolved
- For example, order of stores and loads
- Or each processor may have separate local memory

MP: Message Passing

- Typical MP systems have some way of passing messages between processors.
- This could just be in software using standard I/O facilities
- Or there might be special hardware for the purpose

Implementation Issues

- Does architecture assume specific implementation details
  - Scheduling
  - Caching
  - Pipelining

Compiler Issues

- How is code generated for this machine?
- Any special problems?
- Any special features designed to make translation of specific features easier?

Operating Systems Issues

- How is an Operating System for this machine constructed?
- Any special problems
- Any special features designed to make the life of an OS easier?
  - For example, special instructions for task swiching, state saving, multiple processes accessing virtual memory etc.

End of Lecture

- See you next Tuesday
- Send any problems to class list
  - (see class page for how to sign up)
  - Assignment: send one message to class list
    - Either an interesting question, or just hello!
- These slides will be up on the Web in HTML and Power point formats