Topics in computer architecture

Von Neumann -- The IAS machine

P.J. Drongowski
SandSoftwareSound.net

von Neumann's computer

- "Preliminary discussion of the logical design of an electronic computing instrument," A.W. Banks, H.H. Goldstine, J. von Neumann, 1946
- "The Computer from Pascal to von Neumann, Herman H. Goldstine, Princeton University Press, 1972
- "The Origins of Digital Computers," Brian Randall (editor), Springer-Verlag, 1975
- "Computer Structures: Readings and Examples," D. Siewiorek, C.G. Bell and A. Newell, MacGraw-Hill, 1982 (2nd edition)

General

- Institute for Advanced Studies (IAS)
- Referred to as the "IAS machine"
- Smithsonian Museum
- Application was numerical mathematics
- Vacuum tube electronics (approximately 2,000 tubes)
- Memory
  - Cathode ray storage tube
  - 4096 40-bit word memory
  - 25 microsecond access time
- Speed
  - 0.5 MHz clock rate
  - 20,000 operations per second
- Performed one multiplication in 600 microseconds
- Just for fun - a comparison
  - Atari 800XL
  - $120 (retail price)
  - 6502 single chip processor (3.5 MHz clock)
  - 64Kbytes of memory
  - 344 8-decimal digit multiplies per second
  - 3 millesecond multiplication time

Structure

- Four major units
- Arithmetic unit
  - Binary, parallel arithmetic
  - Decimal too expensive; conversion is simple
  - Performs primitive arithmetic and logical operations
  - Trade-off: Native versus coded operations
    - Native: Fast, requires more hardware
    - Coded: Slow, less hardware in implementation
  - Integer arithmetic only; FP too expensive
- Memory
  - Flip / flop (tube) memory was regarded as impractical
  - Delay lines (dynamic memory) were rejected
  - RCA Selectron tubes
    - 4096 bits per tube
    - 40 tubes in parallel to obtain a word
    - Use "function tables" to address memory
    - Buffer address and data in flip / flop memory
  - Memory was parallel, random access
    - Serial memory is bit at a time and slower
    - EDVAC was serial; used serial delay line memory
- Control unit
  - Compromise between ISA and hardware complexity
  - Instructions consist of:
    - Operation code
    - Memory addresses and mode bits
  - Sequential instruction execution (program counter)
  - Unconditional and conditional branches
    - Needed for decision making
    - Loops for iterative execution
  - System timing derived from a single clock
- Input and output devices
  - Synchronous input / output
  - Programmed data transfer

Instruction set

- Single address per instruction
  - Only small amount of memory is used for addresses
  - Read operand from memory, leave result in AC
- Two instructions stored per 40-bit memory word
  - Provided one instruction lookahead
  - Reduced number of instruction fetch operations
- Basic principle - Incorporate only those features that:
  - Are necessary to have a complete system, or
  - Occur very frequently in mathematical practice

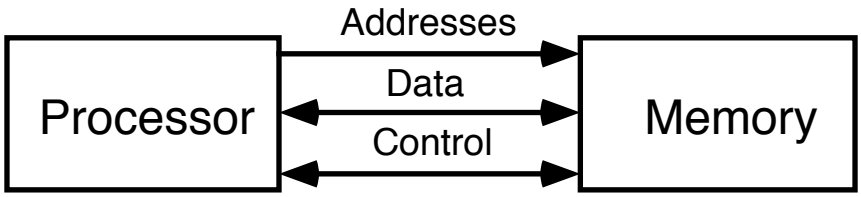
register AC<39:0>, R<39:0>, CC<11:0>	Registers
memory M[4095:0]<39:0>	Primary memory
AC ← M[x]	Load AC from memory
AC ← – M[x]	Load complement
AC ← abs M[x]	Load absolute value
AC ← – abs M[x]	Load negative
AC ← AC + M[x]	Add
AC ← AC – M[x]	Subtract
AC ← AC + abs M[x]	Add absolute value
AC ← AC – abs M[x]	Subtract absolute value
R ← M[x]	Load R from memory
AC ← R × M[x]<78:40> ; R ← R × M[x}<39:0>	Multiply
R ← AC ÷ M[x]; AC ← AC **mod** M[x]	Divide
CC ← M[x]<39:20>	Jump (left - hand)
CC ← M[x]<19:0>	Jump (right - hand)
(AC ≥ 0) ⇒ cc ← M[x]<39:20>	Conditional jump
(AC ≥ 0) ⇒ cc ← M[x]<39:20>	Conditional jump
M[x] ← AC	Store AC to memory
M[x]<39:26> ← AC<39:26>	Store address (left - hand)
M[x}<19:6> ← AC<39:26>	Store address (right - hand)
AC ← AC × 2	Left shift
AC ← AC ÷ 2	Right shift

Programming

- Instructions and data stored together in memory
- Instructions and data are both binary codes
- Instruction format
  - Bits 25 (5) through 20 (0) are opcode / mode bits
  - Bits 39 (19) through 26 (6) are address
- Self-modifying code
  - Instructions can be manipulated as data
  - Instructions can be changed to point to new operands
  - Used to pass subroutine arguments (IAS)
  - Eventually replaced by index registers
  - Modify code through store address instructions
- Arguments against self-modifying code
  - Difficult to debug
  - Cannot have reentrant ("pure") procedures

von Neumann organization

- Most prevelant computer architecture
- Single central processing unit (CPU)
  - Central clock
  - CPU executes instructions in step with clock
  - CPU is the system master
  - Control and communications center for entire system
- Primary memory
  - Linearly addressable array of binary *words*
  - Each word is a fixed size
- CPU - memory *channel*
  - Set of parallel information paths
  - Exchange control signals, addresses, data
  - Sequential operation
    - Set control for read, send address, receive data
    - Set control for write, send address and data
  - Address is an index into the linear memory array
- Programming
  - Instructions and data
    - Binary codes
    - Stored in primary memory
  - Elemental operations on elemental operands
  - Program counter sequences instructions
  - (Un)conditional branches for loops and decisions
  - Arithmetic is parallel binary



Some exceptions

- Burroughs B5500, etc. stack addressable storage
- Burroughs B1800 does not have fixed logical word length
- IBM 370 series has special I/O channels
- CDC Star and Cray have multiple arithmetic processors

Self-consistency (harmony)

- "Recursive machines and computing technology," V.M. Glushkov, et al., IFIP Information Processing '74, North Holland Publishing, 1974.
- von Neumann principles are self-consistent (re-enforcing)
  - Central clock and sequential transfers through channel
  - PC steps sequentially through linear memory array
  - Fixed word width and parallel arithmetic
- Revision of one or two principles will destroy harmony
- Intuitively self-consistency is a "good fit" of principles

Improving von Neumann speed

- Use faster components
  - Increase speed of CPU, memory, channel
  - Bounded by physical limitations
- Exploit locality
  - Keep operands in local cache or general registers
  - Reduces number of channel operations
- Concurrency
  - Increase number of processor - memory channels
    - ◊ Multiple address and data streams
    - ◊ Typically found in vector supercomputers
  - Increase input/output concurrency
    - ◊ Perform computation and I/O simultaneously
    - ◊ Add "direct memory access" channel
    - ◊ Increases speed of bulk data transfers
  - Pipelining
    - ◊ Degree of concurrency limited by number of stages
    - ◊ Pipe disrupted by change in control flow
    - ◊ Disruptions due to data dependency hazards
  - Synchronization and arbitration must be provided
    - ◊ Processes (and devices) must co-ordinate
    - ◊ Interrupts are a synchronization mechanism
    - ◊ Would like to virtualize devices into OS processes

Further problems

- Gap between machine and HL programming language
  - ◊ Implement features of HLL in hardware
  - ◊ Semantic clash
  - ◊ Recursion is usually expensive
- Linear organization does not fit application data model
  - ◊ Leads to construction of sophisticated storage managers
- Sequential centralized control
  - ◊ Complicates operating system design
  - ◊ Processes must be forced into single instruction stream
  - ◊ Nondeterministic effects must be suppressed
  - ◊ Inhibits extensibility
- Throughput depends on component speed
  - ◊ Wire delay is now dominant
  - ◊ Compounded by chip partition

Recursive machine principles

- No limit on complexity of operators and operands
  - ◊ Operands may be numbers, strings, vectors, whatever
  - ◊ New types recursively mapped to primitive structures
  - ◊ Implies a "procedural" data model (i.e. executable data)
- Program elements executed when operands are available
  - ◊ Use memory associativity to detect "ready" elements
- Memory structure is reprogrammable
  - ◊ Convert data and programs to internal form
  - ◊ New types and machines are recursively definable
  - ◊ One level implemented in terms of other levels
- No limit to number of machine elements
  - ◊ Performance tuning by adding and removing elements
  - ◊ Replace broken machines dynamically
- Organization is flexible, re-programmable
  - ◊ Communication switches needed between RC elements
  - ◊ Switches are programmable
  - ◊ Note: An RC element is a processor-memory pair