Modern Computer Architecture

Lecture 4
History
• Overview of Computer Architecture
  – Turing, von Neumann
  – Babbage, Zuse
  – Moore’s Law
  – Microprogramming
  – IBM-360
  – Cray, Connection Machine, Transputer
  – 6502 → 68000, 8080 → 80286
  – CISC, RISC
  – VLIW, Super-Scalar
• Based on Hilbertsches Programm:
  – precise formal language
  – theorem proving on finite methods

• Turing machine: emulates a mathematician
  – based on formalism to verify proofs
  – hyp: “intelligence is the ability to perform mathematics”
  – Universal Turing Machine: can emulate any other Turing Machine

$$(s', m'(p), p') = f(s, m(p), p)$$
Immediate Predecessors

• Colossus (1943)
  – electronic calculator made from tubes
  – 5,000 characters/s (paper tape with 12.2 m/s)
  – used to break German “Enigma” code during WW2
  – design top secret, no manuals, all machines dismantled until 1960
  – Alan Turing worked on this project
  – hardwired program (Turing machine, but not “universal Turing machine”)

• Atanasoff-Berry-Computer (1937-1941)
  – Fully electronic calculation (vacuum tubes)
  – binary arithmetics, ALU
  – separate computation and memory
  – purpose: linear equation system solver
  – not programmable
  – US District Court for the District of Minnesota ruling 1979: first computer
  – Influenced construction of ENIAC
Failed Projects

- Charles Babbage: Analytical Engine
  - first plan of Turing Machine
  - mechanical
  - did not work 1837
  - 3 minutes for multiplication

- Konrad Zuse: Z3
  - first working universal Turing machine
  - electro-mechanical (relays)
  - <10 seconds for multiplication
  - scrapped by NS war regime
First Computers

• Harvard Mark I (1944)
  – electromechanical design
  – 16m x 2.4m x 61cm, 4.5t, 4 kW
  – 6s per multiplication
  – No branches, but Turing complete
  – reliable compared to tubes
  – 5PS motor for mechanical synchronization

• ENIAC (1946)
  – electronic design, digital, decimal system
  – 30 m x 2.4 m x 90 cm, 27t, 159 kW
  – 0.2 ms per addition, 2.8 ms per multiplication
  – Turing complete
  – first use: artillery firing tables for the US Army
  – 17,468 vacuum tubes, 70,000 resistors, ~5 million hand-soldered joints
• Fundamental Principles of Computer
• Manual of computer for University of Pennsylvania (EDVAC)
• data and program stored at the same address space
• conscious engineering realization of Turing machine
- idea: store function for Turing machine in memory (ROM or RAM)
- realize phases of instruction execution
  - fetch instruction
  - decode instruction
  - read operands
  - execute instruction
  - write result

Applications
- IBM 360 commercial computers
- Motorola 68000 and Intel x86
- Intel Processors have loadable microcode today (used for bug fixing)
- Nintendo-64 and Playstation-2
- GPU programs?
Computers Made From Discrete Transistors

- **Minuteman D-17B (1962)**
  - built from diodes and transistors
  - 28kg, 345kHz
  - 2985 word harddisk, no RAM

- **Apollo Guidance Computer (1969)**
  - 61x32x17cm, 32kg, 55W
  - 4,100 ICs, each containing a single 3-input NOR gate
  - 2kB RAM, 36kB ROM, 2MHz
  - OS: “exec” runs 8 threads
- Very little memory available, memory consumption is paramount
- Memories about 10x slower than logic
- Instruction memory bandwidth is scarce resource at this time!
- ISA was interpreted by μ-code
- Complex instructions due to sophisticated decoding techniques

- **6502**: consumer processor (C64), employed accumulator ISA (similar to stack based)
- **x86: 8085/8086** extends accumulator architecture to (non-orthogonal) register architecture, FPU **8087** maintains stack architecture
- **VAX** has orthogonal ISA for register architecture, complex instructions
- **68000** is a more efficient, less orthogonal register architecture
- **Cray**’s vector processors also have complex ISA
- **Transputer** (T800) has compact stack based ISA

**Terminology used:**
- **orthogonal**: in a register architecture, every architectural register can be used for any operation
IBM/360 architecture

- IBM enterprise computing
- still in use today IBM z-Series
- backward compatible architecture
- micro coded

influences:
- 8 bit bytes
- byte addressable words á 32 bit
- super-scalar out-of-order execution (model 91)
Digital Equipment (DEC) VAX

- “mini computer”, division computing
- PDP-8 to PDP-11
- Virtual Address Extension (VAX)
- VMS: predecessor of Windows NT
- development platform for Unix
- VAX 11/780: reference for 1 MIPS (million instructions per second)
- orthogonal instruction set architecture
- CISC instruction set with micro code
for (i=0; i<n; i++)
  c[i] = a[i] + b[i];

\[ c[0:n-1] =_{\text{vector}} a[0:n-1] +_{\text{vector}} b[0:n-1]; \]

for (i=0; i<n; i+=s)
  c[i:i+s-1] =_{\text{SIMD}} a[i:i+s-1] +_{\text{SIMD}} b[i:i+s-1];

- Cray-1: s=64
- Intel: s=16 (AVX)
Single Instruction, Multiple Data (SIMD) according to Flynn
A large set of processing elements (PEs) execute a single stream of instructions on different sets of data
Overhead for storing and fetching instructions is reduced
Different processing elements are synchronized

Examples:
- MMX, ISSE (s=4)
- Cray (s=64)
- Control Data CDC (mem-mem)

Startup overhead
- Connection Machine CM-1 (1983)
- 65,536 processors á 1-bit
- Popular for image processing
- also optimized Lisp and sparse matrix operations
• **Scan-Operation:**
  - Apply operation sequentially to elements of a vector
  - Result is a vector of element-wise intermediate results
  - Example: $A \rightarrow \text{scan}(A,+)$

\[
A = [\begin{array}{cccccccc}
5 & 1 & 3 & 4 & 3 & 9 & 2 & 6 \\
0 & 5 & 6 & 9 & 13 & 16 & 25 & 27 \\
\end{array}]
\]

• **Segmented Scan:**
  - new parameter: segment descriptor/flags
  - “restart” operation at beginning of each segment

\[
B = [\begin{array}{cccccccc}
5 & 1 & 3 & 4 & 3 & 9 & 2 & 6 \\
0 & 5 & 0 & 3 & 7 & 10 & 0 & 2 \\
\end{array}]
\]

\[
S_b = [\begin{array}{cccccccc}
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 5 & 0 & 3 & 7 & 10 & 0 & 2 \\
\end{array}]
\]
Applications of Segmented Scan

- Segmented Scan is inherently a sequential operation
- It can be implemented efficiently even on highly parallel architectures (GPU, Connection Machine, Cray, etc.)
- Segment flags can be used to store irregular structures, like graphs and lists.
- Parallelization of Algorithms:
  - quick-sort, radix-sort
  - graph: e.g. minimum spanning-tree
  - sparse matrices
• Chuck Peddle MOS Technologies
• 3510 transistors (nMOS)

• Applications
  – Commodore PET 2001, VC-20, 64
  – Atari 800 XL
  – Apple I, Apple II
  – BBC Micro
  – Super-Nintendo
Motorola 68000

- 16/32-bit processor architecture influenced by VAX
- influential in home computer era
- micro coded

- Jack Tramiel: Commodore → Atari
- Jay Miner developed Amiga multi-media chipset
• long history of backward compatibility:
  – 8008: first 8-bit integrated processor
  – 8080/8085: 8-bit mainstream
  – 8086: 16-bit extension
  – 80286: memory protection
  – 80386: 32-bit extension
  – 80486: 1CPI RISC
- Exploit thread level parallelism
- Put several computers with communication capabilities on a PCB each
- many threads, no shared memory
- Programming language for communicating threads: Occam
Parallel Algorithms

Data parallelism: (e.g.: SIMD)

Functional parallelism: (e.g.: pipelining)
Pipelining Instruction Execution

- Functional Parallelism
  - instruction fetch
  - instruction decode
  - execute
  - memory access
  - write back
Memories were getting substantially larger, while memory speed was approaching the speed of logic.

Compilers for VAX used only a subset of the available instructions

VAX implementors accelerated mostly used instructions

Effect: Larger programs using instruction subset were running faster than shorter programs exploiting the ISA

**Idea of RISC:**

- Goal: One clock cycle per instruction (CPI), parallelism through pipelining
- And: Offload as much work as possible from hardware to compiler

**Measures:**

- Regular instruction encoding for fast decoding, abandon μ-code (fixed size instructions)
- Load/Store Architecture (complex instructions can not be executed in a single cycle)
- Orthogonal ISA (single type of registers)
- Large number of registers (to aid compiler with register assignment problem)

**Disadvantages:**

- Larger code size
- Difficult to program in assembly
Why is RISC more difficult to program in assembly?

- No more addressing modes: every value had to be loaded from memory explicitly, and has to be assigned into a register
- No more composite instructions for common tasks, e.g., bit field manipulation, looping constructs, address modification, string handling, BCD arithmetic, stack management
- Assembly code is unreadable, it mainly consists of move, add, branch instructions
- Assembly programmer has to keep track of register assignments
- Assembly programmer has to take care of low-level administrative tasks, like saving registers and stack frame creation at procedure entry and exit
- Most of the traditional assembly speedup strategies have become useless [M. Abrash, 80x86 Optimization, Dr. Dobb’s Journal, Mar. 1991]
Symbolic Computing

- Computers optimized for symbolic processing
- Complex memory management
- Garbage Collection
- Complex micro programs

- 1987 David Ungar: “Smalltalk on a RISC” ➔ no necessity to build specialized hardware for symbolic processing
Stanford RISC

- Processor pipeline is visible to compiler
- **MIPS** means “*microprocessor without interlocked pipeline stages*”
- Assembly language has to take care about potential pipeline hazards
- Compiler can predict hardware behavior more easily
- Branches have delay slots
- Delay for value assignments (MIPS: load instruction)

Berkeley RISC

- Processor pipeline is invisible to the compiler
- Possible change pipeline internals
- Pipeline hazards are detected by hardware and lead to stalls
- Hardware is more complex
- Examples: **SPARC**, AMD **29k**
- Facilitates to development of superscalar architectures

 mv r1, #1
 st r1, 8(sp)
 mv r2, #2
 ld r2, 8(sp)
 add r3, r2, #10
 add r4, r2, #10

r3 will be 12
r4 will be 11
• Predecessors:
  – CDC 6600 supercomputer (1964) by Seymour Cray

• Started 1981 by John Hennessy at Stanford Univ.
  – 1985: R2000
  – 1988: R3000

• Silicon Graphics Supercomputers
  – Distributed Shared Memory (NUMA)

• Nintendo-64, Playstation, Playstation-2

• Chinese Clone: Loongson
• Designed by Sophie Wilson and Steve Furber
• influenced by 6502 (low latency)
• first silicon 1985
• 32-bit RISC microprocessor with 30,000 transistors
• built for Archimedes home computer

• first embedded use:
  Apple Newton
• 98% market share mobile phones
  (1 bn. units, 2005)
• 90% market share for embedded 32-bit

• VLIW architecture (parallel instructions for Control, ALU, Shifter)
Almost all processors became 1 CPI
Challenge: execute more than one instruction per cycle

VLIW (very long instruction word)
- Pipeline is visible to compiler
- Code density (very long instruction words)
- Inflexible at changing pipelines and dynamic situations (caches)

IBM VLIW
Texas Instruments C6x
Intel IA-64 (EPIC)

Superscalar
- Pipeline is invisible to compiler
- Use hazard detection to launch independent instructions in parallel
- Effort in hardware
- Can react flexibly (cache misses)

MIPS R10k
DEC Alpha
x86
VLIW (very long instruction word)

• Representatives:
  – Elbrus (1986) ➔ SPARC ➔ Transmeta
  – IBM VLIW (1986) ➔ multi-chip, issues with interconnect
  – Intel EPIC (explicitly parallel instruction computer)
    • Intel Itanium IA-64
  – Texas Instruments (TI) C6x DSP
DEC Alpha

- 1992
- 64-bit
- very simple pipeline
- extreme clock speed
Figure 1. In CPU architecture today, heat is becoming an unmanageable problem.
(Courtesy of Pat Gelsinger, Intel Developer Forum, Spring 2004)