ECE 471 – Embedded Systems
Lecture 2

Vince Weaver

https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

6 September 2024

Announcements

- Reminder: The class notes are posted to the website.
- HW#1 was posted today (Friday), will send e-mail

Review From Last Time – Embedded Systems Characteristics

- Embedded
- Fixed-purpose
- Resource Constrained
- Lots of I/O
- Real-time Constraints

What does Embedded Hardware Look Like?

- Anything from 8-bit/tiny RAM to 64-bit 2GHz 2GB
- Performance has greatly improved over the years.

Type			Speed	RAM	Disk	GPU
Intel	Xeon	64-bit	4GHz	16GB	1TB	Nvidia
ARM	A53	64-bit(?)	1GHz	1GB	8GB	VC4
ARM	M0	32-bit	32MHz	16kB	128kB	none
MOS	6502	8-bit	1MHz	64kB	140kB	none

Embedded System or Supercomputer?

- Modern embedded boards are as fast as early supercomputers
- Cray2 (1985)=1.9 GFLOPS
- Pi2 = 1.5 GFLOSP
- Pi3 = 5 GFLOPS
- Pi4 = 13 GFLOPS
- Pi5 = 34 GFLOPS
- and this isn't even counting the GPU

Processor Types / ISAs

- Intel/AMD x86/x86_64 (mostly in desktop/laptop/server)
- ARM (extremely common in embedded)
- RISC-V (newish, has relatively open licensing)
- Older RISC systems: Power, MIPS, SPARC
- Older CISC systems: m68k, VAX
- Older 8-bit: AVR, 8051, 6502, z80, 6809, 68HC11
- Many many more

Embedded Systems 20 years ago

- Somewhat dated list, from EE Times 2003. Multiple answers so doesn't necessarily sum up to 100%

- 8-bit processors
  - Microchip PIC – 43%
  - AVR, etc. 8051 – 55%
  - Motorola 68xx – 36%
  - Zilog Z80 – 15%

- 16-bit processors
  - 8086/80186/80286 – 41%
  - 68HC12 – 21%

Are 8-bit Systems Still Used?

- One popular example is the Arduino with 8-bit AVR ATmega
- Also, some chips like 8051 were popular for years (still found in many USB devices), so legacy systems still around at companies that need to be maintained.

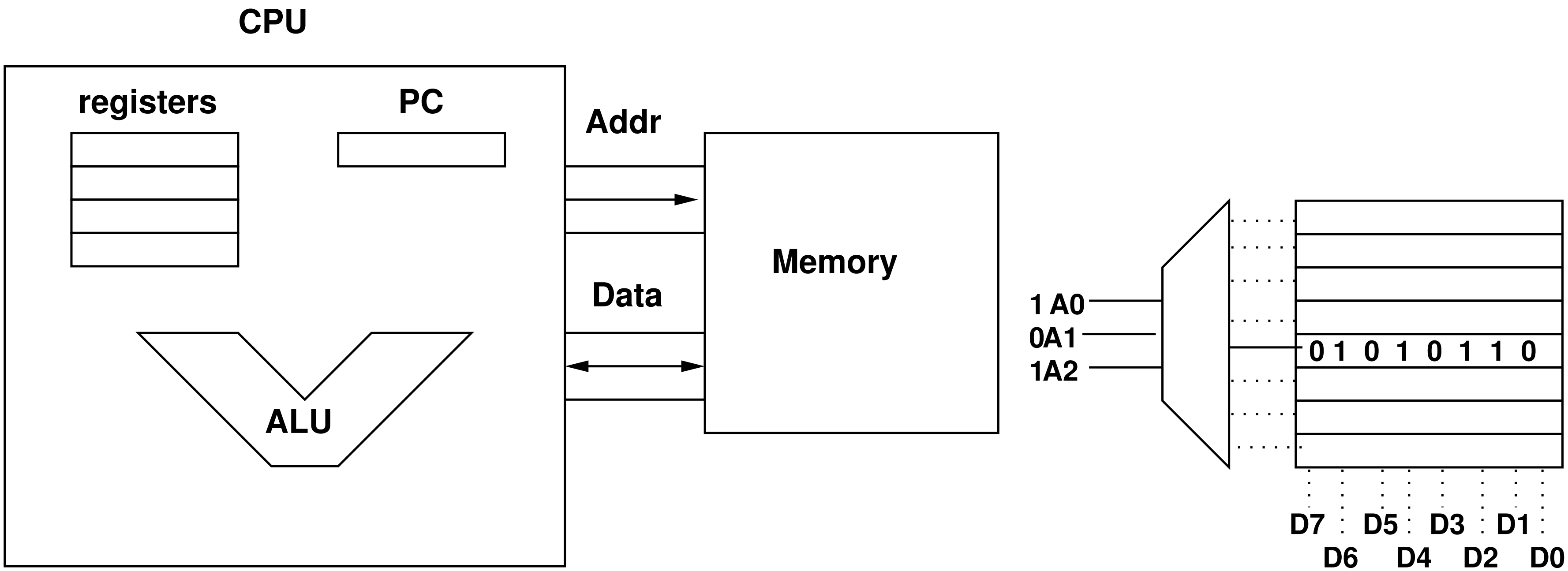
We'll Mostly Use ARM in this Class

- Widely used
- You'll see if it you move to industry
- Other classes in ECE using it
- There is some concern recently, mostly due to the uncertainty of their parent company (failed purchase by NVIDIA) and licensing costs
- Maybe one day will be supplanted by RISC-V

Microprocessors

- First one considered to be 4004 by Intel (for use in calculator, 1971)
- First to include all of a CPU on one chip. Before that there were processors, but often were made out of many discrete chips (sometimes entire boards full of logic)

Simple 8-bit CPU Block Diagram



Modern computers are more or less incomprehensible and essentially magic. You can take ECE473 or ECE571 to learn more about how they work.

What makes a processor 8-bit vs 16-bit vs 32-bit vs 64-bit?

- The size of the registers?
- The size of the address bus?
- The size of the data bus?
- The size of the ALU (integer math unit)?
- The size of the PC (program counter)?

Believe it or not giant video-game system flamewars happened over these questions.

Answer Not Always Clear

- On modern systems it typically is the integer registers, as well as the maximum size of a memory pointer (which typically is the same as the integer register size)
- On many systems though it is not as clear cut.

8-bit Systems

- A "pure" 8-bit system would have 8-bit registers (0-255), 8-bit ALU, and an 8-bit data bus.
- However an 8-bit address bus (only 256 bytes of RAM) is too limiting so most 8-bit processors (6502, z80, 8080, etc) had 16-bit address busses, 16-bit PCs, and often 16-bit register capability

16-bit Systems

- Most 16-bit processors were equally complex.
- The 8086 had 16-bit registers and 16-bit data bus, but a 20-bit address bus with complex addressing.
- To complicate things, the 8088 was 8086 compatible but had only an 8-bit data bus (to save cost, with the side effect of making memory accesses take twice as long)

32-bit Systems

- Most 32-bit processors have 32-bit registers and 32-bits of address space, but that limits to 4GB
- Some have extensions (x86 and ARM) allowing 36-bits of address space.
- Data bus has been made complex by caches and are often quite large
- Often there are larger registers on chip (64-bit or 80-bit floating point, 128-bit SSE, 256-bit or 512-bit AVX)

64-bit Systems

- Most 64-bit processors have 64-bit registers, but their address bus is often limited (to 36 - 40 bits, 48-bits, maybe 56-bits nos this is complicated by virtual memory)
- It was always a problem of programmers stealing top bits of pointers as being "unused" only to be sad later when things got bigger (ARM26, IBM, macos/m68k)
- A few recent chips have "ignore top bits" option to allow this (ARM: TBI (top bit ignore), Intel LAM (Linear Address Masking))

Other Possibilities?

- 128-bit systems? RISCV has a spec
- Do machines have to be a power-of-two in bitness? No, not necessarily. 36-bit machines were once quite popular.

Microcontroller

- Sometimes abbreviates MCU (micro controller unit) or μC
- Microcontroller was generally a small CPU for use in embedded systems
- You'll still hear the term used
- Sometimes will be used specifically for low-end embedded systems

System-on-a-Chip / System-on-Chip

- Moore's law allows lots of transistors
- Discrete Chips: CPU, GPU, Northbridge, Southbridge, (and older days, FPU, MMU, etc)
- System-on-a-Chip (SoC): All parts of computer on-chip CPU, DSP, memory, timers, USB, voltage regulators, memory controllers
- System-in-Package (SiP): various chips in one package

Extra Features on SoCs

- Parallel and Serial I/O
- A/D, D/A converters
- GPIO pins
- i2c, CAN, SPI, 1-wire, USB busses
- FPGA?
- Low-power
- Sound, DSP
- Video, GPU, Video Codecs
- Timers, PWM

Dedicated Hardware vs Programmable

- ASIC – Application Specific Integrated Circuit
  direct wiring of state machines / logic on silicon die
- FPGA – reprogrammable low-level logic
- Microcontroller – can do what above do, but in software
- Why use ASIC: could be faster, but what if mistake?
  Why use FPGA: could be faster, more expensive/complex
  Why use microcontroller: Cost. Time to market. Bug-fixes (easier to fix in software)

Tradeoffs

It's all about tradeoffs

- Power
- Performance
- Cost
- Compatibility
- Time to Market
- Features

Challenges vs Regular Systems

- Programming in constrained environment (cross-compiling?)
- Security
- Safety
- Real-time
- Power consumption
- Long-life (embedded device might be in use for decades)
- Testing
- Bug-fixing

Discussion

- What concerns might you have when designing an embedded system?
  Security is a big one these days
- What language might you write your code in?
  C is still popular despite security issues.