

ECE 571 – Advanced Microprocessor-Based Design Lecture 4

Vince Weaver

`http://web.eece.maine.edu/~vweaver`

`vincent.weaver@maine.edu`

9 September 2020

Announcements

- Homework #1 was posted, due Friday
- Let me know if you have any trouble logging in
- Yes, sometimes computer architecture research is a lot of boring measurements.

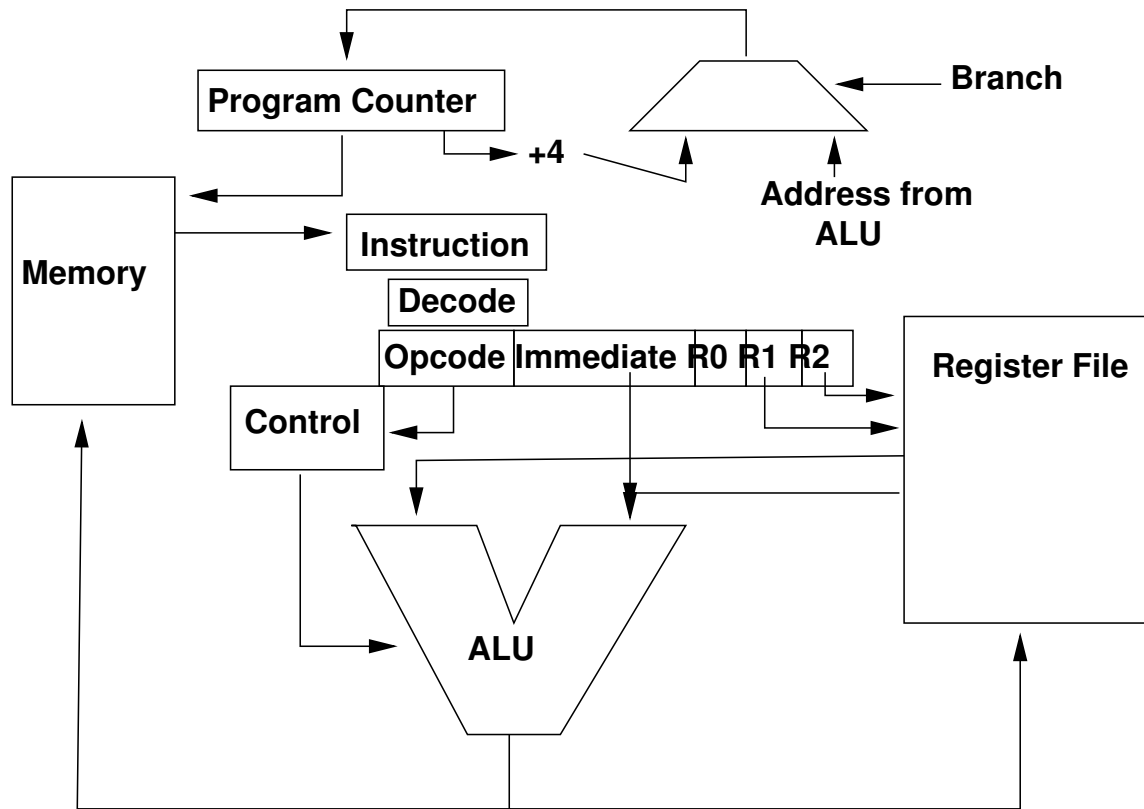


Microprocessors

- Also known as Central Processing Unit (CPU)
- Do the general purpose calculations in a system
- Originally big, multi-cabinet, multi-board, multi-chip
- The first “micro” processor fit on one chip.
Often regarded as the 4-bit Intel 4004. (history?)
- In the old days you could buy a discrete CPU, plop onto circuit board, hook up some memory and a terminal, and you had a computer.
- These days things are a lot more complex.



Simple CPU



Simple CPU – Breakdown

- Program Counter / Instruction Pointer points to next instruction
Increments each clock and loads next instruction from memory. If a branch instead loads new address if branch taken.
- Instruction is decoded. Opcode (says what type of instruction), Registers to use, possibly an immediate value.
- Opcode goes to control (usually a PLA) or microcode



that splits up signals to all the functional units and tells them what to do (what kind of operation, whether to read or write, to branch or not, etc)

- Source register values are read from register file and fed to ALU
- ALU does math/logic based on control
- Result written back to destination register on register file.
- If load or store instruction, then address calculated (often by ALU) and sent to memory. If load, value written to reg file, if store value to write sent out



- Once instruction is done, advance to next instruction.



CPU – Design decisions

- ISA (Instruction Set Architecture) – what insns do you need?

ALU: add/sub/and/or/xor (mul and divide? many do not have)

memory: load/store

branch: branch if zero/not zero

shift?

nop? often a pseudo-op

syscall?



compare (can be implemented with subtract)

Lots of other stuff *can* be added. Floating Point. String copy. Predicated/conditional execution. Crazy polynomial/vector insns.

- Other decisions: how many arguments to opcode? 2 or 3?
- Flags register?
- Number of registers? 1/3/8/16/32/128/windowed?
- Decode logic: Fixed-width 4-byte (32-bit) instructions?
Completely variable sized instructions (x86 1-15 bytes?)
VLIW (3-instructions in a 128-bit package?)



Embedded (THUMB, THUMB2) mostly 16-bit (Code Density)

- RISC vs CISC
- Big or Little Endian?
- Bitsize (4, 8, 16, 32, 64 bit?)

What does that mean? Size of registers? ALU? Memory Address Range? Data bus width? Complex issue.



Design Constraints

- Number of pins
 - DIP (dual inline package): 4004 = 16,
z80/6502/8080/8086 = 40
 - PGA (pin grid array): Pentium = 273
 - LGA (land grid array) pins on socket not chip):
Sandybridge = 1000+

