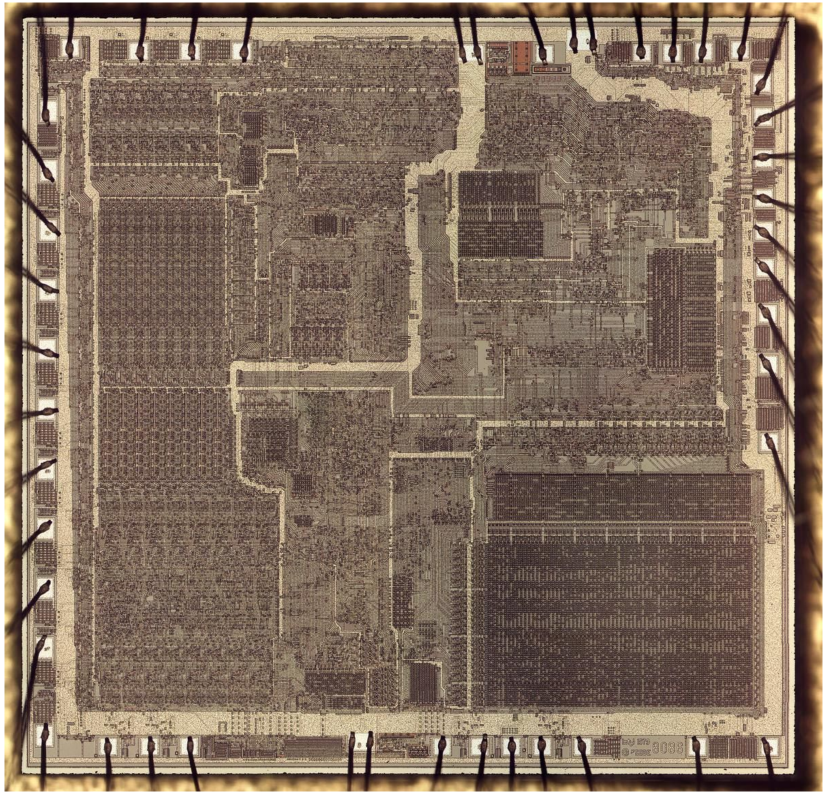
MONTANA
STATE UNIVERSITY

The CPU

The Heart of the Computer

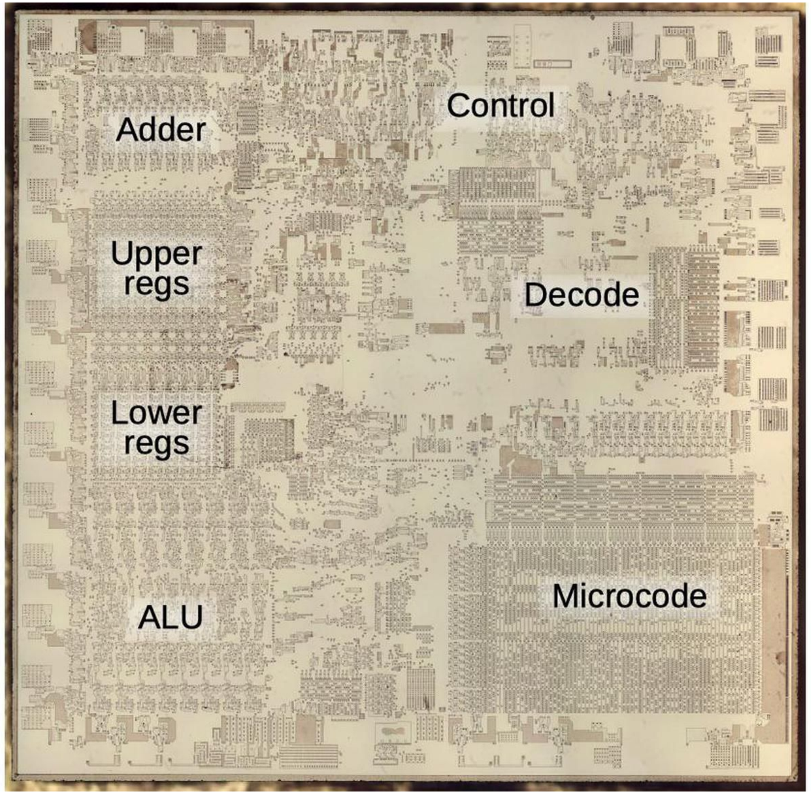
The CPU

- For the next two lectures we are going to be considering the *CPU*
  - Central Processing Unit
- Pictured at right is the 8086 CPU



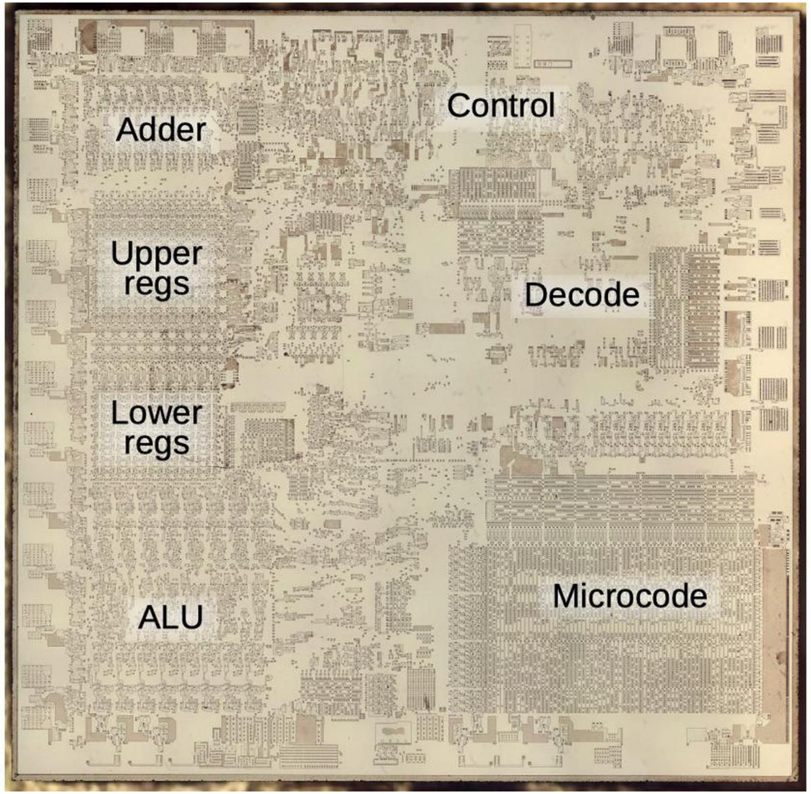
The CPU

- Major components of the chip
  - Adder (for memory, we will ignore)
  - Registers
  - Control Unit
  - Decoding logic
  - Microcode
  - Arithmetic Logic Unit (ALU)



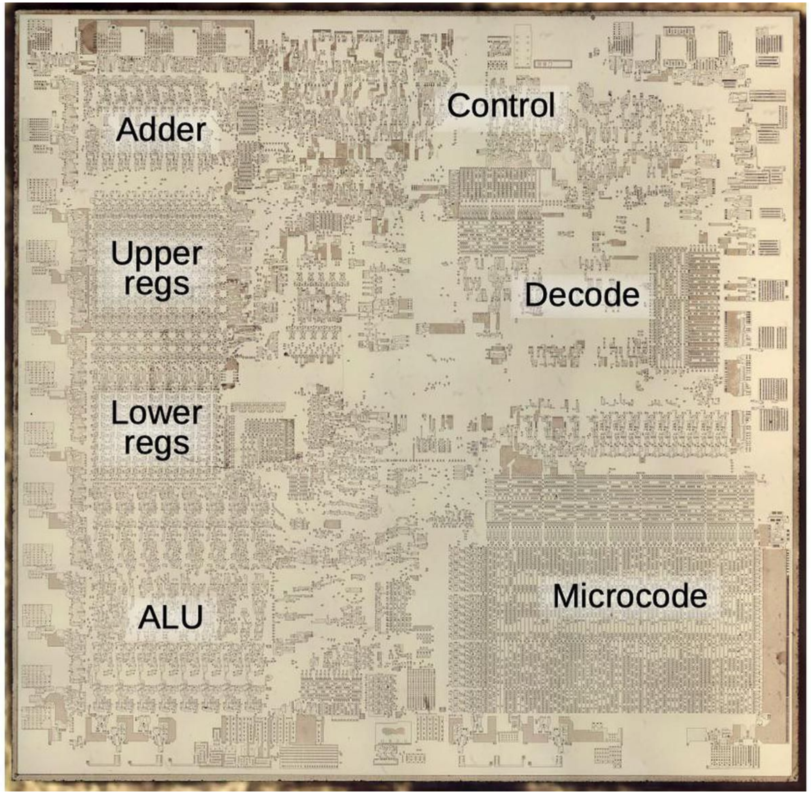
The CPU

- Registers
  - We have already discussed what registers are
  - Very fast memory located close to the CPU
  - Can interact with the ALU and one another quickly



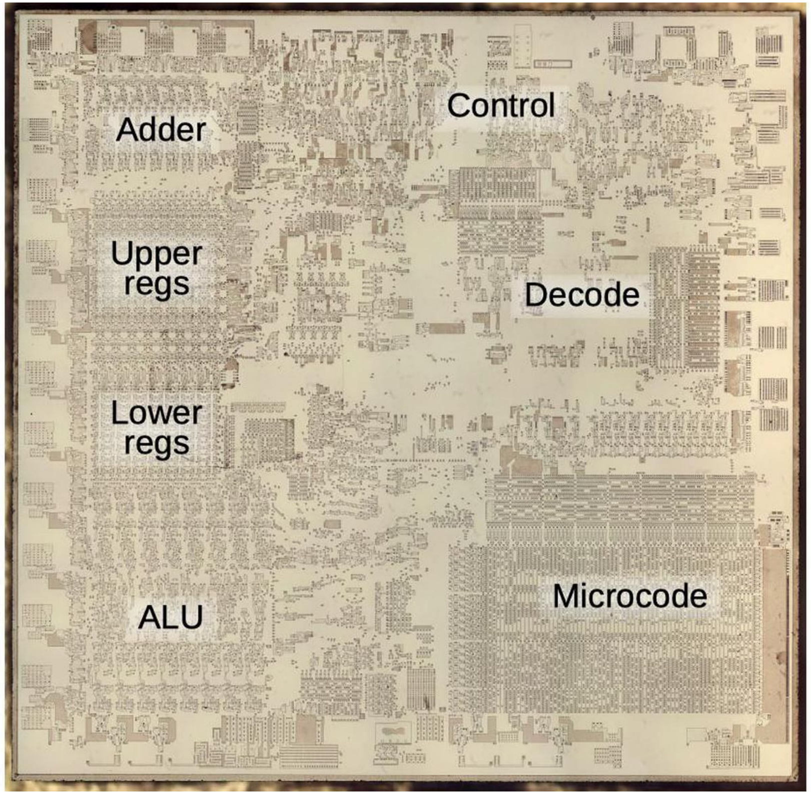
The CPU

- Control Unit
  - Directs the operation of the processor
  - “Tells” the memory, ALU and other devices how to respond to instructions being executed by the CPU



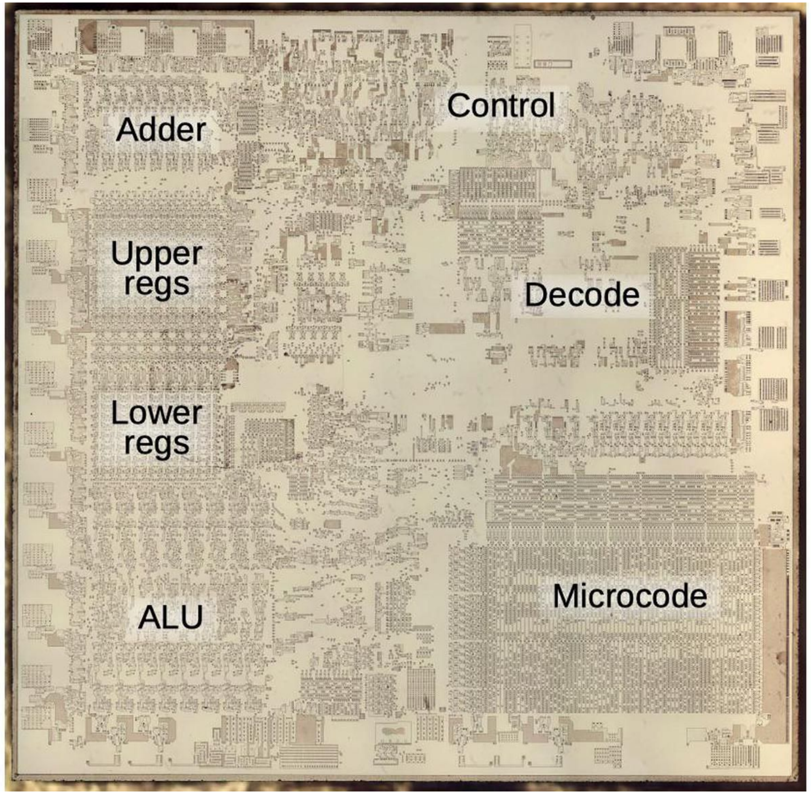
The CPU

- Decoding Logic
  - Instruction are “decoded” into a series of control unit operations
  - Feeds the control unit with what to do for a given instruction
  - This is all done via logical gates based on the concepts we have discussed earlier



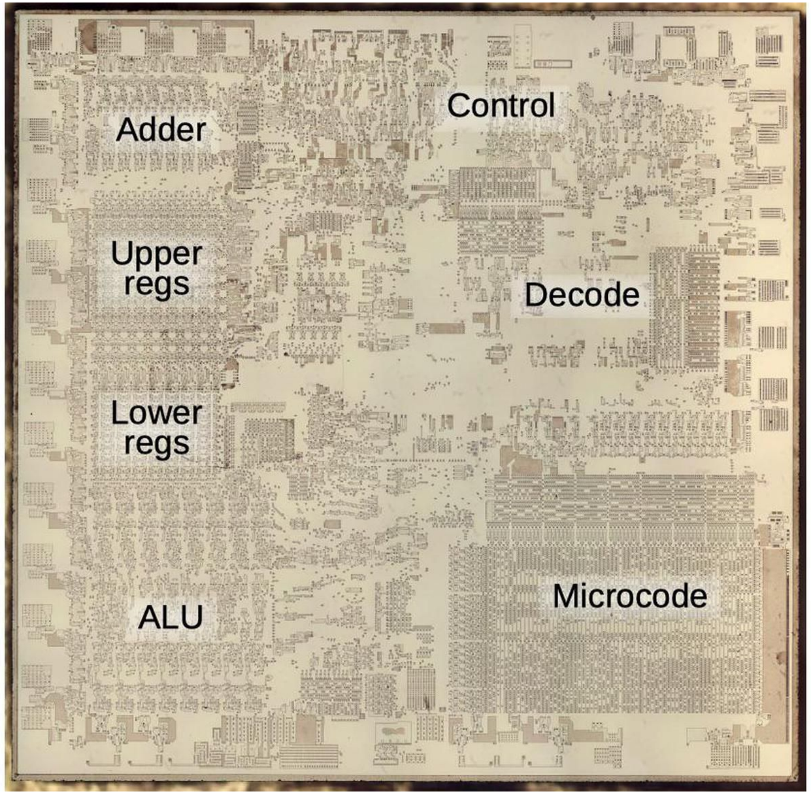
The CPU

- Microcode
  - Encoded logic for each instruction
  - Recall the x86 is a CISC architecture
  - With microcode, complex instructions can be reduced to a series of simpler instructions embedded in logic on the chip



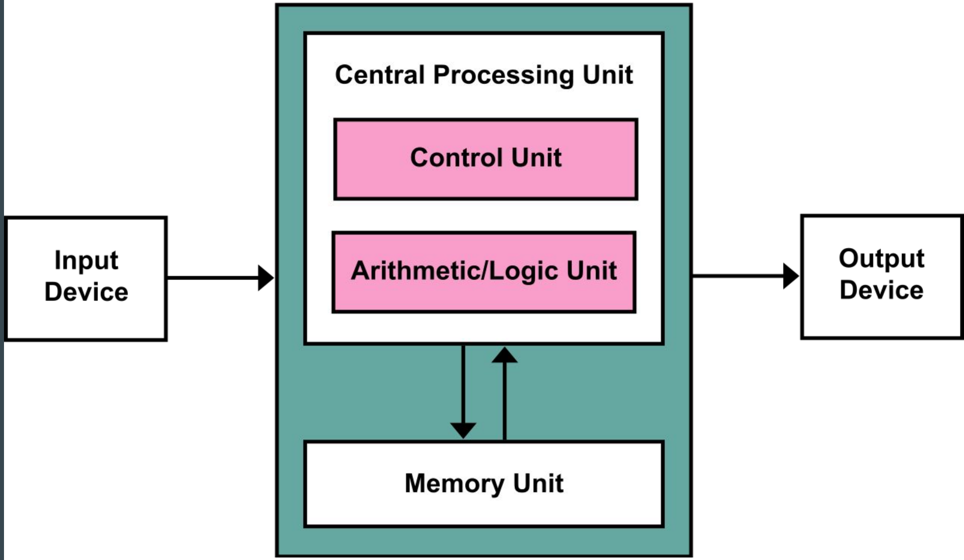
The CPU

- The ALU
  - Arithmetic Logic Unit - does most of the CPU work
  - Contains circuitry like the adder we looked at previously
    - More sophisticated than a simple ripple carry model
    - Also contains operations like shift, compare, etc
  - Also contains temporary registers, etc.



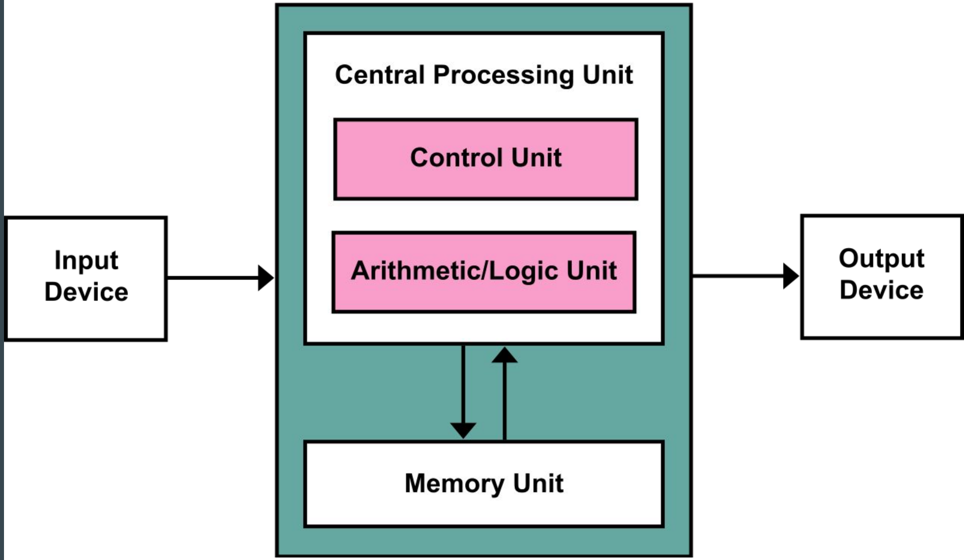
The Von Neumann Arch

- This chip is an example of the *Von Neumann Architecture*
    - CPU with a control unit and ALU
        - Control unit contains a program counter and instruction register
    - Memory via a BUS
        - Stores data *and* instructions
    - I/O mechanisms



The Von Neumann Arch

- Note that instruction fetch and data fetch cannot be in parallel
  - They share a common bus with the memory unit
  - This is known as the *Von Nuemann Bottleneck*



John von Neumann

- Born December 28, 1903 in Budapest, Kingdom of Hungary
  - Born to a non-observant jewish family
- Child prodigy in mathematics
  - Photographic memory
  - Produced a significant mathematical paper per month for first three years of academic career

John von Neumann

- Eventually found his way to the Institute of Advanced Study at Princeton
    - Einstein and others complained about von Neumann’s preference for loud german marching music

John von Neumann

- Significant contributions in
  - Pure Mathematics
  - Set Theory
  - Geometry
  - Quantum Mechanics
  - Quantum Information Theory
  - Game Theory
  - Economics
  - Fluid Dynamics
  - The Manhattan Project

John von Neumann

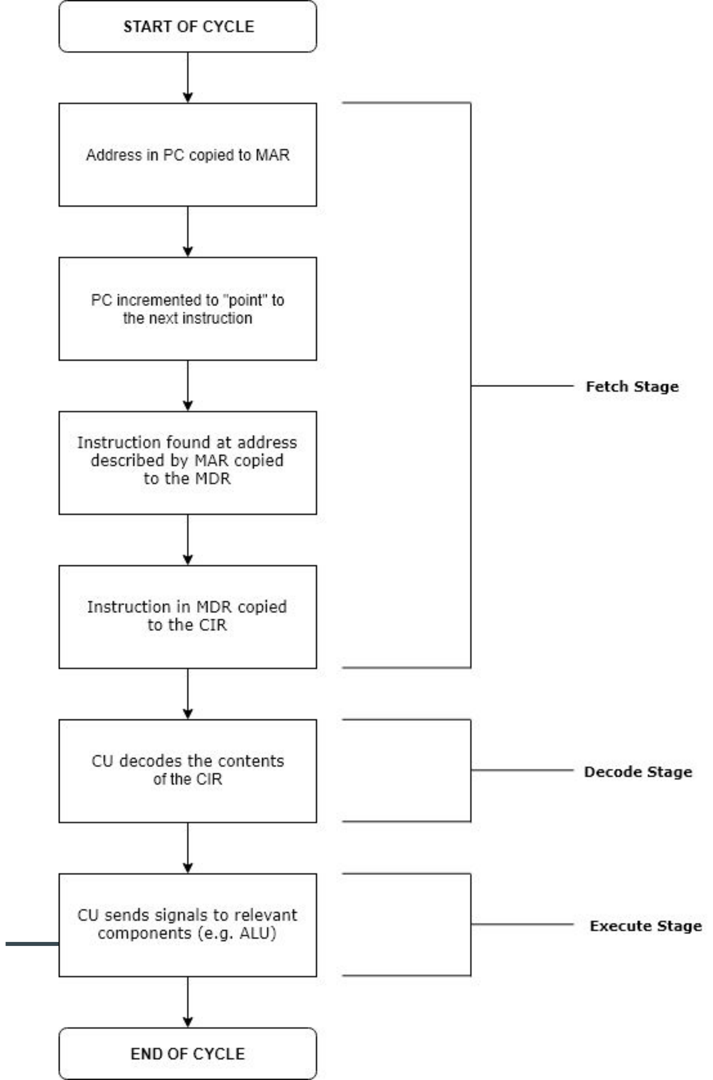
- Oh, right, he also was the primary author of the First Draft of a Report on the EDVAC
  - Laid out the basic architecture that we are talking about and using today
  - Back before semiconductors existed

John von Neumann

- John von Neumann is my intellectual hero
- If you are looking for an intellectual hero, you could do a lot worse that this guy

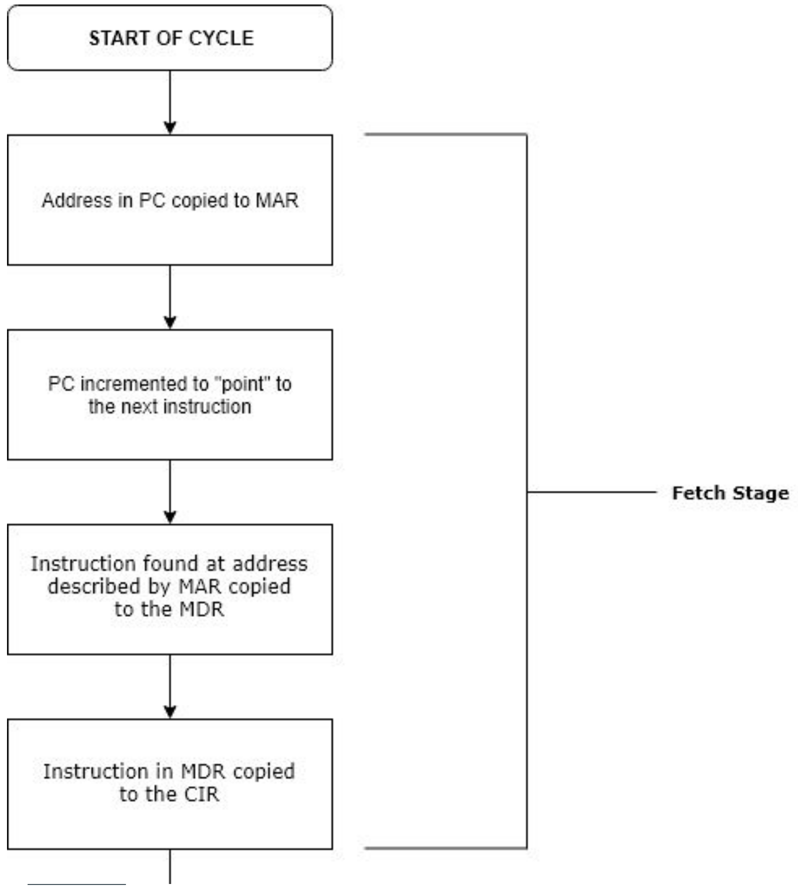
The Fetch-Execute Cycle

- The basic cycle of the CPU in the Von Neumann architecture is called the fetch-execute or fetch-decode-execute cycle
    - This cycle can be executed serially or, in more modern CPUs, in parallel
- The following is a rough description of each step


START OF CYCLE
Address in PC copied to MAR
PC incremented to "point" to the next instruction
Instruction found at address described by MAR copied to the MDR
Instruction in MDR copied to the CIR
Fetch Stage
CU decodes the contents of the CIR
Decode Stage
CU sends signals to relevant components (e.g. ALU)
Execute Stage
END OF CYCLE

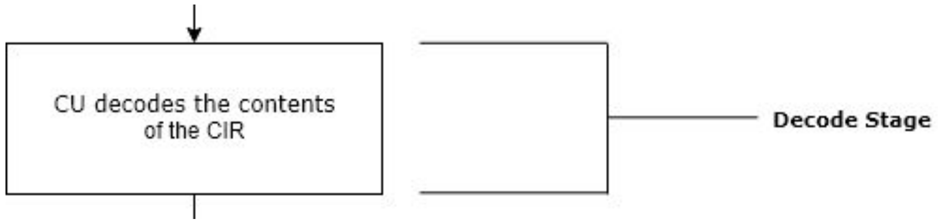
The Fetch Stage

- Fetch Stage Steps
    - Address in Program Counter register is copied to the MAR
        - Memory Address Register
    - Program Counter is incremented to next instruction position
    - Instruction found at the address is copied into the MDR
        - Memory Data Register
    - The MDR is copied to the CIR
        - Current Instruction Register


START OF CYCLE
Address in PC copied to MAR
PC incremented to "point" to the next instruction
Instruction found at address described by MAR copied to the MDR
Instruction in MDR copied to the CIR
Fetch Stage

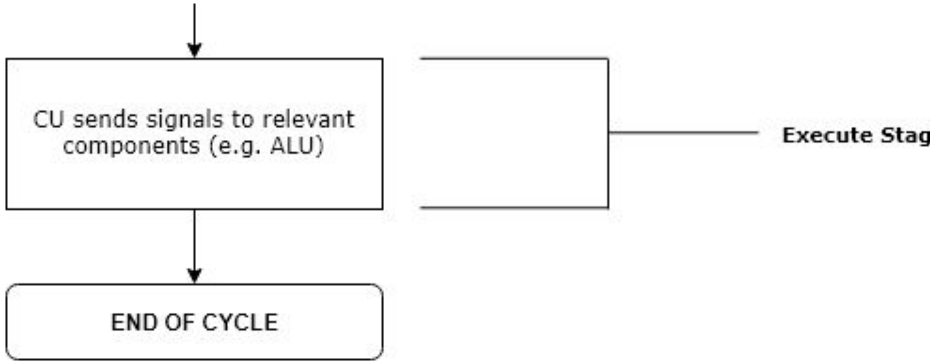
The Decode Stage

- Decode Stage Steps
  - The Control Unit (or a sub-component, the Decode Unit) takes the instruction in the CU and determines what to do
  - Sends signals to the appropriate circuitry in the ALU to activate the correct logic
    - NB: a memory read may stall execution in this step to load a new value into the MDR



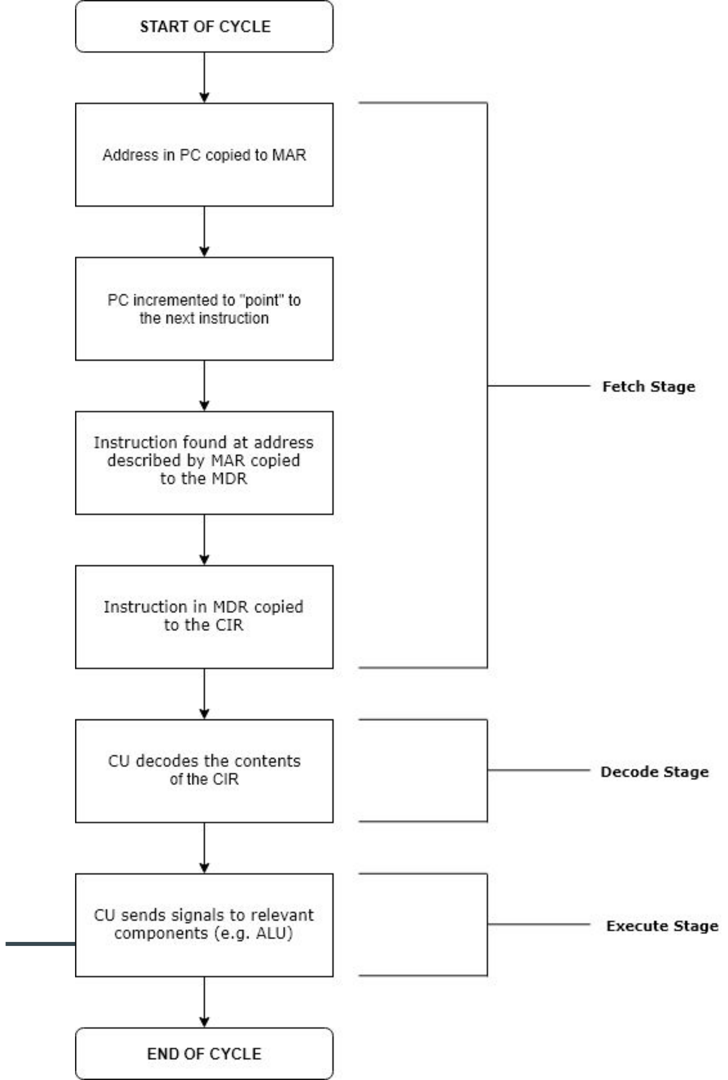
The Execute Stage

- Execute Stage Steps
  - Based on signals received from the control unit after it has decoded an instruction, the ALU performs mathematical or logical functions on data stored in registers
  - ALU writes values to locations (registers or memory)
  - May update the PC to point to a new location



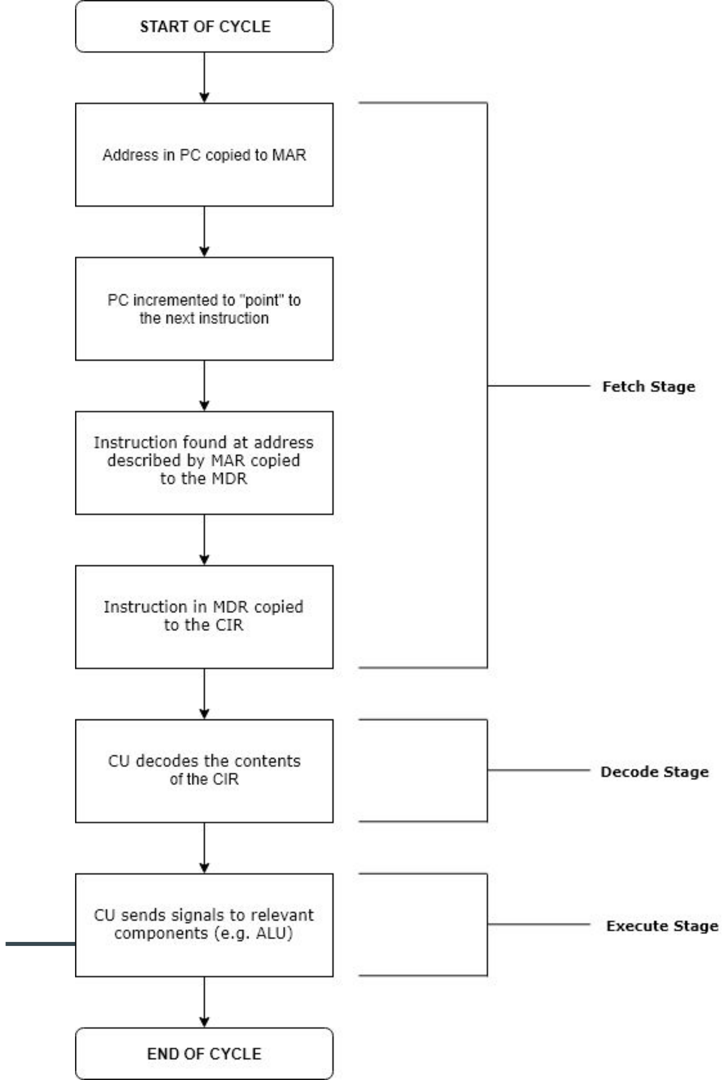
The Execute Stage

- Repeat Cycle
- Now repeat that really, really fast
    - Like a few billion times per second fast


START OF CYCLE
Address in PC copied to MAR
PC incremented to "point" to the next instruction
Instruction found at address described by MAR copied to the MDR
Instruction in MDR copied to the CIR
CU decodes the contents of the CIR
CU sends signals to relevant components (e.g. ALU)
END OF CYCLE
Fetch Stage
Decode Stage
Execute Stage

The Execute Stage

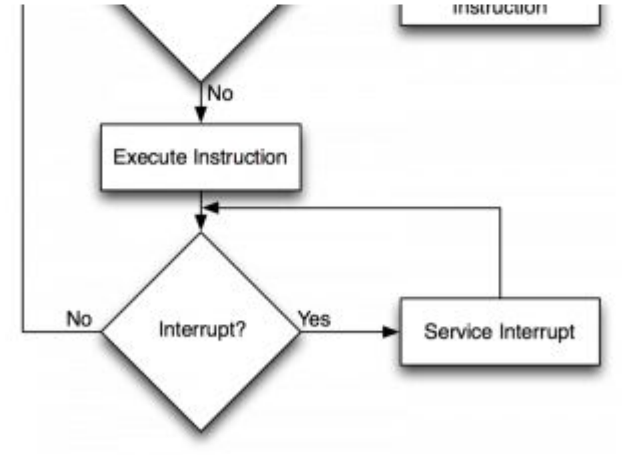
- Repeat Cycle
- Now repeat that really, really fast
    - Like a few billion times per second fast



START OF CYCLE
Address in PC copied to MAR
PC incremented to "point" to the next instruction
Instruction found at address described by MAR copied to the MDR
Instruction in MDR copied to the CIR
CU decodes the contents of the CIR
CU sends signals to relevant components (e.g. ALU)
END OF CYCLE
Fetch Stage
Decode Stage
Execute Stage

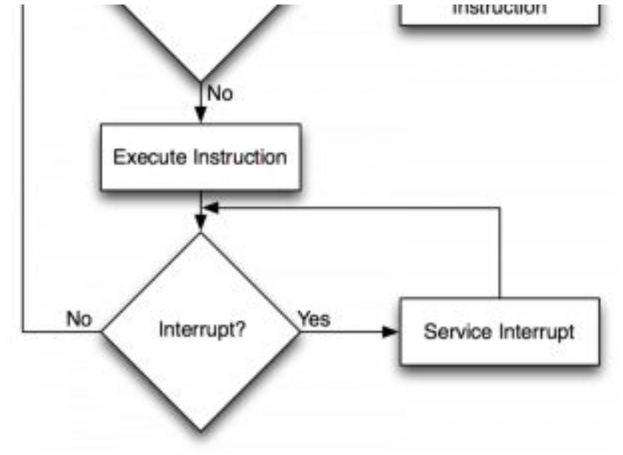
The Interrupt Stage

- I/O devices need a way to signal to the CPU that input is ready
- This is done with an *interrupt*
- Interrupt handling forms another stage in the modern CPU cycle



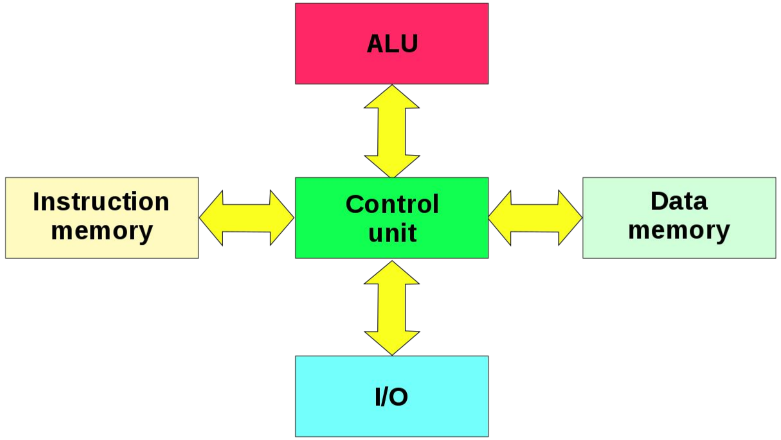
The Interrupt Stage

- Also used by the operating system to *time share* the CPU across multiple processes
  - A “Timer Interrupt” runs the OS schedule, which must pick the next process to be run on the CPU



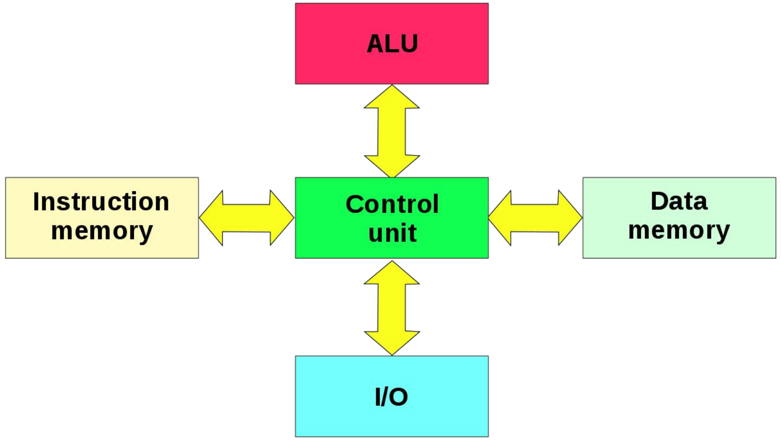
The Harvard Architecture

- The Von Neumann architecture is not the only possibility
- Early on *The Harvard Architecture* was a competing layout
    - Separate pathways for instructions and data



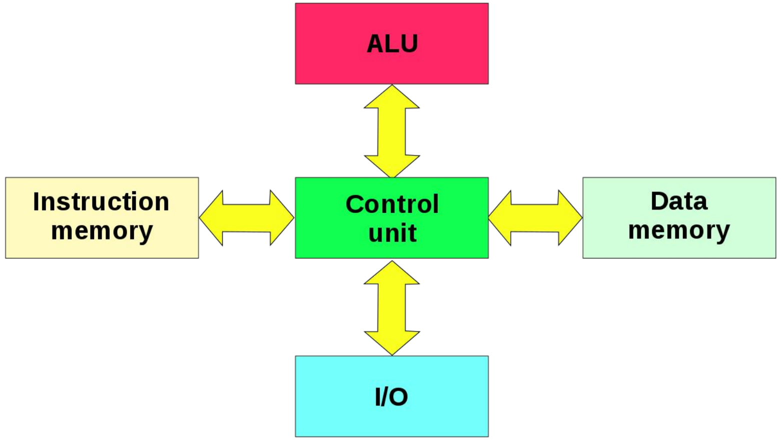
The Harvard Architecture

- Modern CPUs have moved towards this architecture by introducing separate caches for instructions and data
    - Sometimes called a *Modified Harvard Architecture*



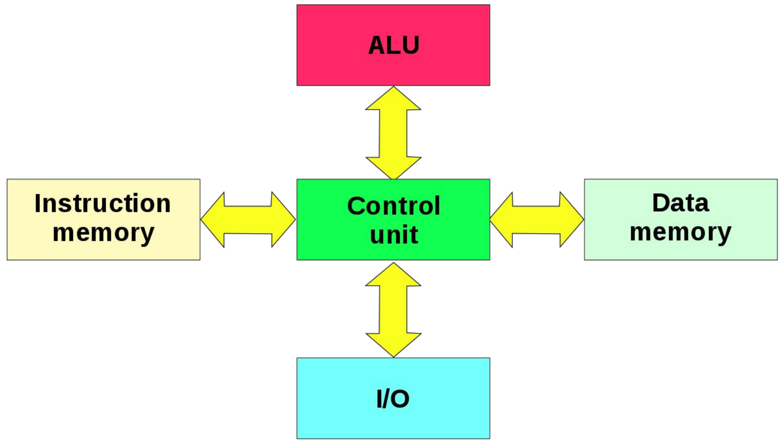
The Harvard Architecture

- When instructions are retrieved from an instruction cache, a CPU acts like a Harvard Architecture
- When instructions must load from memory, the CPU acts like a Von Neumann Architecture



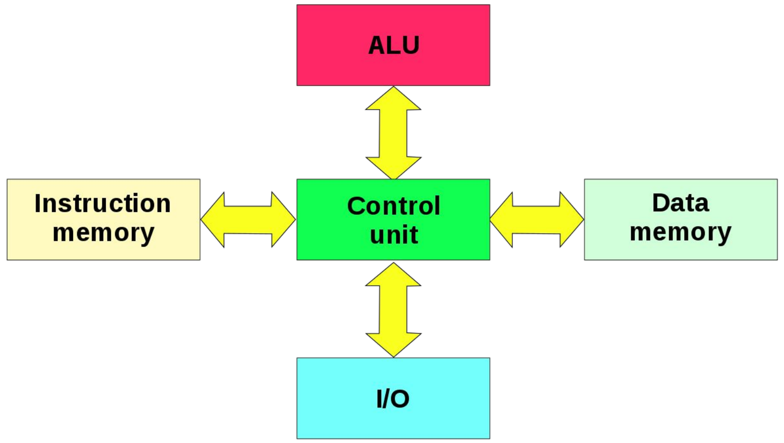
The Harvard Architecture

- This is typically hidden from the user/programmer
- To us, the programmer, modern machines look like a Von Neumann Architecture, even if internally it usually acts like a Harvard Architecture device



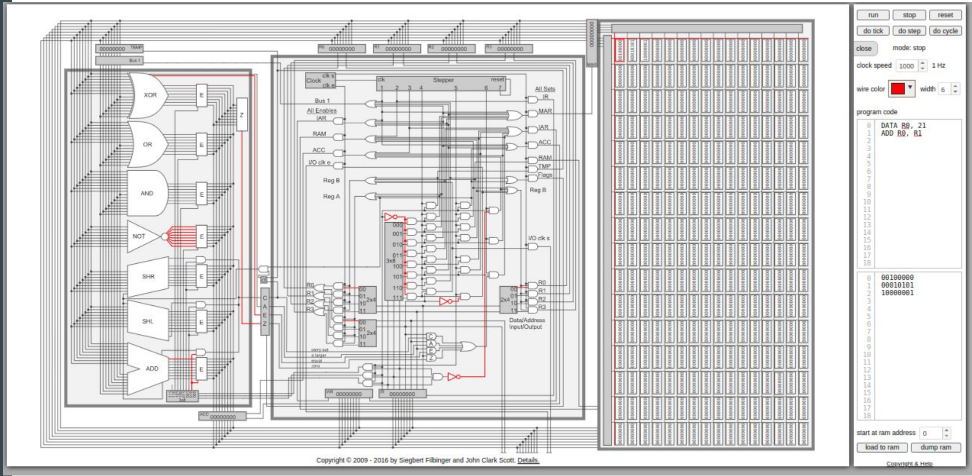
The Harvard Architecture

- This is typically hidden from the user/programmer
- To us, the programmer, modern machines look like a Von Neumann Architecture, even if internally it usually acts like a Harvard Architecture device



The Scott CPU

- In our next session we are going to be discussing the Scott CPU and each step in the Fetch/Execute cycle using a simple 8-bit model
- Very similar to the LMC and RISC simulators we looked at earlier!



mode: stop

clock speed 1000 1 Hz

wire color 🟥 width 6

program code

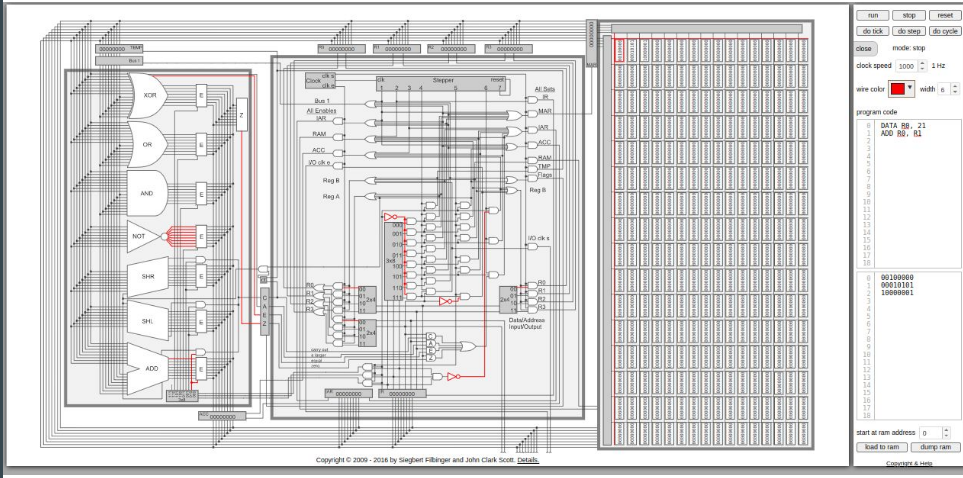
```
DATA R0, 21
ADD R0, R1
```

```
00100000
00010101
10000001
```

start at ram address 0

The Scott CPU

- But with a LOT more detail
- We will drill in to
  - The ALU
  - The Control Unit
  - Memory & Registers
- The emulator is available here: http://buthowdoitknow.com/but_how_do_it_know_cpu_model.html



mode: stop

clock speed 1000 1 Hz

wire color 🟥 width 6

program code

```
DATA R0, 21
ADD R0, R1
```

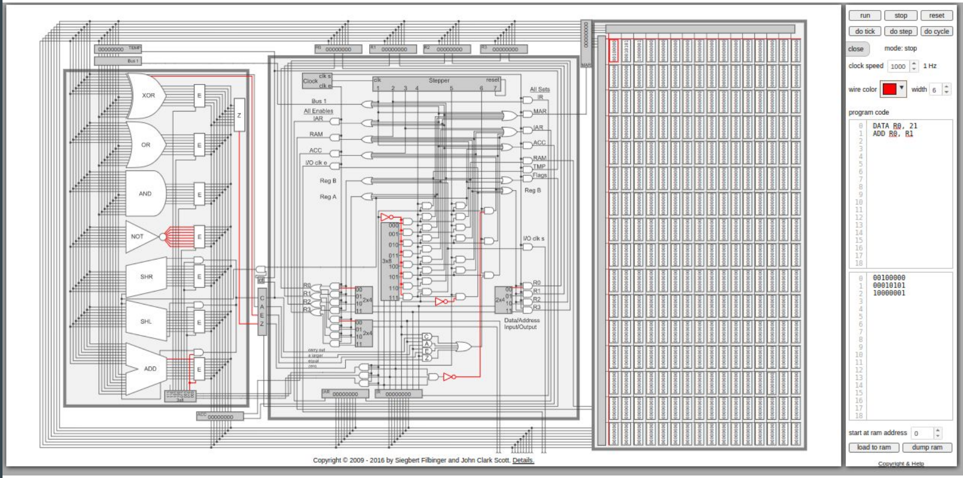
```
00100000
00010101
10000001
```

start at ram address 0

load to ram · dump ram

The Scott CPU

- Scott CPU Assembly Language
- As you might expect, the Scott CPU has a simple assembly language
- Note: The Scott CPU is 8 bit
    - Has only 4 registers
    - Has only 256 memory slots

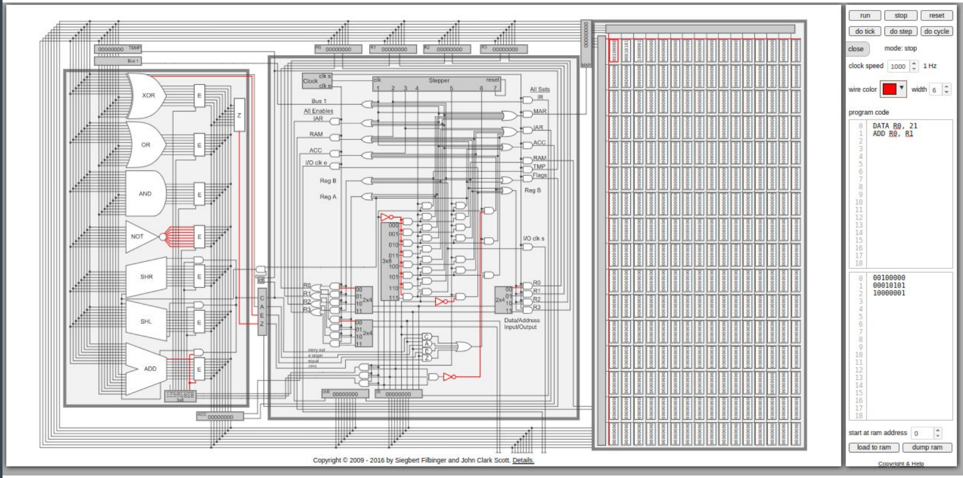


```
DATA R0, 21
ADD R0, R1
```

```
00100000
00010101
10000001
```

The Scott CPU

- Some Scott CPU Assembly instructions
  - ADD, etc. <R1>, <R2>
  - AND, etc. <R1>, <R2>
  - CMP <R1>, <R2>
  - LD, ST <R1>, <R2>
    - Load memory pointed to by R1 to R2
  - DATA <R1>, <Value>
    - Load Value into R1
  - JA, JZ, JE
    - Jump if greater, zero or equal



clock speed 1000 1 Hz

width 6

```
DATA R0, 21
ADD R0, R1
```

```
00100000
00010101
10000001
```

start at ram address 0

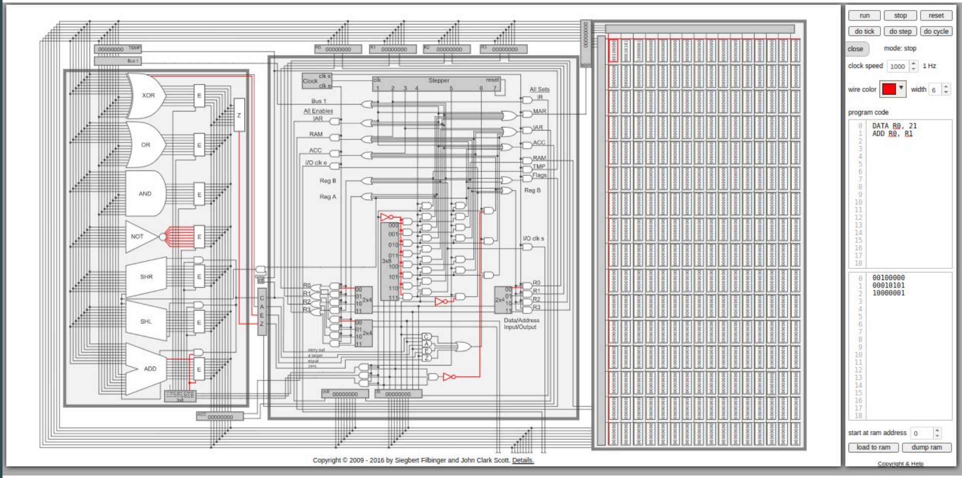
Copyright © 2009 - 2016 by Siegbert Filbinger and John Clark Scott. Details.

The Scott CPU

- Great preview video

  https://www.youtube.com/watch?v=cNN_tTXABUA

- Please watch as prep for our next lecture!



clock speed 1000 1 Hz

wire color 🟥 width 6

program code

```
DATA R0, 21
ADD R0, R1
```

```
00100000
00010101
10000001
```

start at ram address 0

Copyright © 2009 - 2016 by Siegbert Filbinger and John Clark Scott. Details.

The CPU

- The CPU is the core component of a modern computer
- It consists of various components
    - Control Unit
    - ALU
    - Registers
- Most CPUs use the Von Neumann Architecture
    - With the common fetch, decode, execute, (interrupt), repeat loop
- The Harvard Architecture lost initially, but has come back in modified form, via internal, hidden to the programmer, caches
- Next time we will drill in on implementation details of the CPU

MONTANA
STATE UNIVERSITY