1945: John von Neumann

- Wrote a report on the stored program concept, known as the First Draft of a Report on EDVAC
- also Alan Turing… Konrad Zuse… Eckert & Mauchly…

The basic structure proposed in the draft became known as the “von Neumann machine” (or model).

- a **memory**, containing instructions and data
- a **processing unit**, for performing arithmetic and logical operations
- a **control unit**, for interpreting instructions
von Neumann Model

Abstraction of von Neumann

inputs:
- Keyboard
- Mouse
- Scanner
- Card reader
- Disk

output:
- Monitor
- Printer
- LED
- Disk

memory:
- stores both program instructions and data

processing unit:
- ALU
- TEMP

control unit:
- PC
- IR

implements a next state function
Memory

$2^k \times m$ array of stored bits

Address
  - unique ($k$-bit) identifier of location

Contents
  - $m$-bit value stored in location

Basic Operations

LOAD
  - read a value from a memory location

STORE
  - write a value to a memory location
Interface to Memory

How does processing unit get data to/from memory?

**MAR**: Memory Address Register

**MDR**: Memory Data Register

**To LOAD** a location (A):
- Write the address (A) into the MAR.
- Send a “read” signal to the memory.
- Read the data from MDR.

**To STORE** a value (X) to a location (A):
- Write the data (X) to the MDR.
- Write the address (A) into the MAR.
- Send a “write” signal to the memory.
Hardware Level Organization

Major components:
- memory
- central processing unit
- registers
- the fetch/execute cycle
  (the hardware process)

PC = program counter
IR = instruction register
MAR = memory address register
MBR = memory buffer register
I/O AR = I/O address register
I/O BR = I/O buffer register
Central Processing Unit

Control
- decodes instructions and manages CPU’s internal resources

Registers
- general-purpose registers available to user processes
- special-purpose registers directly managed in fetch/execute cycle
- other registers may be reserved for use of operating system
- very fast and expensive (relative to memory)
- hold all operands and results of arithmetic instructions (on RISC systems)
- save bits in instruction representation

Data path or arithmetic/logic unit (ALU)
- operates on data
Stored Program Concept

Instructions are collections of bits
Programs are stored in memory, to be read or written just like data

memory for data, programs, compilers, editors, etc.

Fetch & Execute Cycle

Instructions are fetched and put into a special register
Bits in the register "control" the subsequent actions
Fetch the “next” instruction and continue
Of course, on most systems several programs will be stored in memory at any given time.

On most contemporary systems instructions of only one of those will be executed at any given instant.

The operating system will rapidly switch among the eligible processes, producing the illusion that several programs are executing at the same time.
Sometimes called the hardware process… executes continuously.

Steps:
- fetch an instruction from memory to the instruction register
- increment the program counter register (by the instruction length)
- decode the instruction (in the control unit)
- fetch operands, if any, usually from registers
- perform the operation (in the data path); this may modify the PC register
- store the results, usually to registers
Finite State Machine

So, where's the FSM?

- Input devices
- Contents of registers and memory
- Current State
- Output devices
- Execute Instruction
- Next Instruction
- Input
Control Unit State Diagram

Here's a state machine diagram for the MIPS control logic we'll be studying: