

Control Units: Hardwired & Microprogrammed

Lecture for CPSC 5155

Edward Bosworth, Ph.D.

Computer Science Department

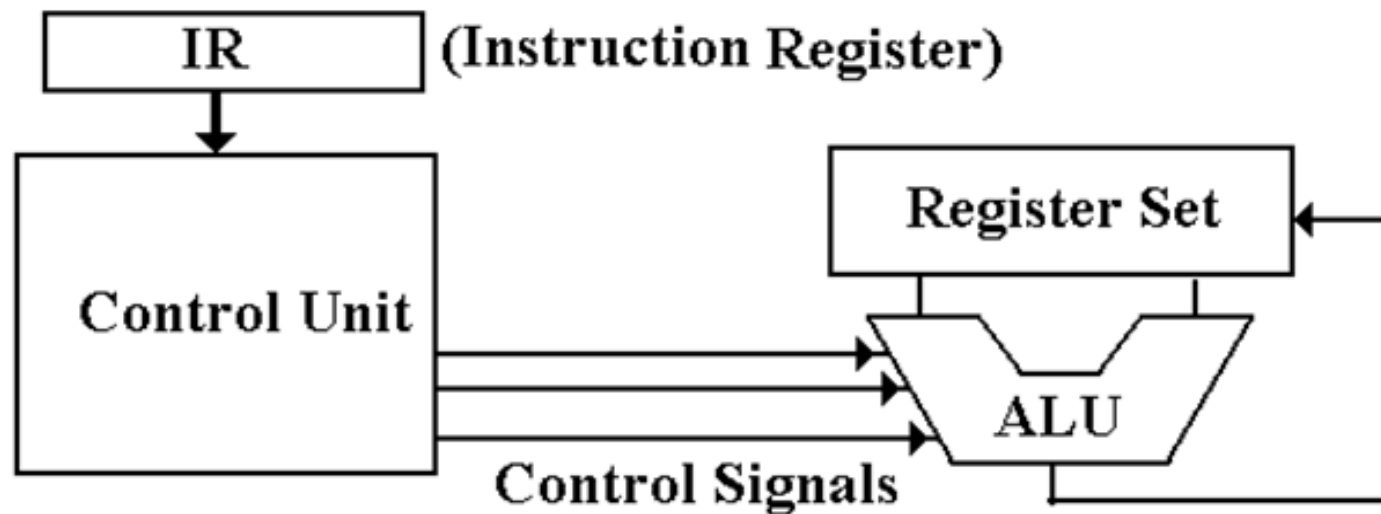
Columbus State University

The Central Processing Unit (CPU)

- The CPU has four main components:
 1. The Control Unit (along with the IR) interprets the machine language instruction and issues the control signals to make the CPU execute that instruction.
 2. The ALU (Arithmetic Logic Unit) that does the arithmetic and logic.
 3. The Register Set (Register File) that stores temporary results related to the computations. There are also **Special Purpose Registers** used by the Control Unit.
 4. An internal bus structure for communication.

The Control Unit (Part 2)

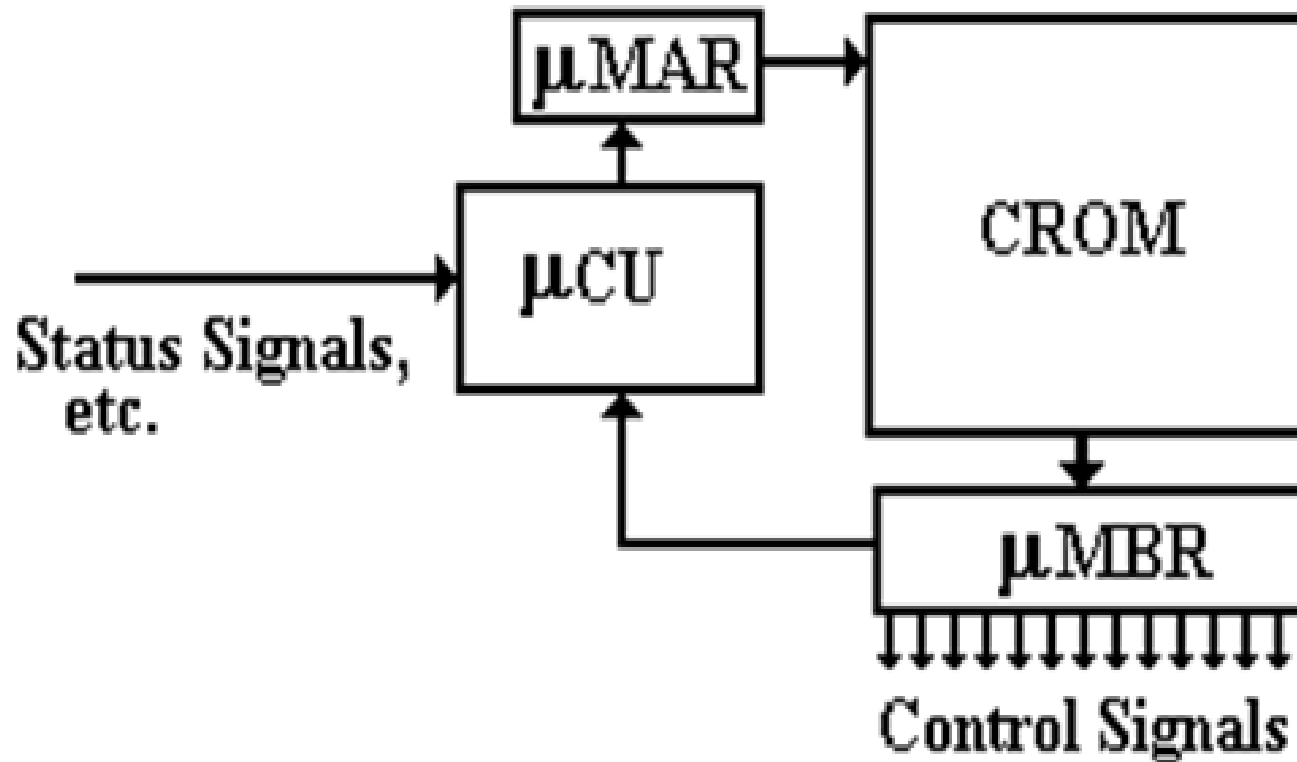
- The function of the **control unit** is to decode the binary machine word in the IR (Instruction Register) and issue appropriate control signals. These cause the computer to execute its program.



Two Options for the Control Unit

- **Hardwired:** The control signals are generated as an output of a set of basic logic gates, the input of which derives from the binary bits in the Instruction Register.
- **Microprogrammed:** The control signals are generated by a microprogram that is stored in **Control Read Only Memory**.
- The **microcontroller** fetches a control word from the **CROM** and places it into the **μMBR**, from which control signals are emitted.

The Microprogrammed Control Unit



Design of the μ CU

- The only function of the **micro-control unit** (μ CU) is to compute the address of the CROM word next to be placed into the μ MBR.
- As such, it is extremely primitive and simple.
- Even for a large sophisticated computer, design of the μ CU might be suitable as a project for undergraduate students.
- This simplicity makes the unit very attractive.

The Boz-5: A Very Simple Computer

- We shall use some material from a didactic computer designed by your instructor to illustrate various control unit designs.
- In this design, instruction execution is divided into three phases. We focus on the first of these phases: **fetch**, which fetches the next instruction and updates the PC.

The Common Fetch Sequence

- In the Boz-5 design, the Fetch sequence is divided into four phases, each having duration of one clock pulse.
- Here are the microoperations associated with the first three phases of the fetch sequence.
- Step 1: $(PC) \rightarrow MAR, READ.$
- Step 2: $(PC) + 4 \rightarrow PC.$
- Step 3: $(MBR) \rightarrow IR.$

Control Signals are More Primitive

- Control signals directly enable transfers, so they must be very low level.
- Note that the inputs (Fetch, T0, T1, T2) are discrete binary signals.

Fetch, T0: (PC) \rightarrow B1, **tra1**, B3 \rightarrow MAR, READ.

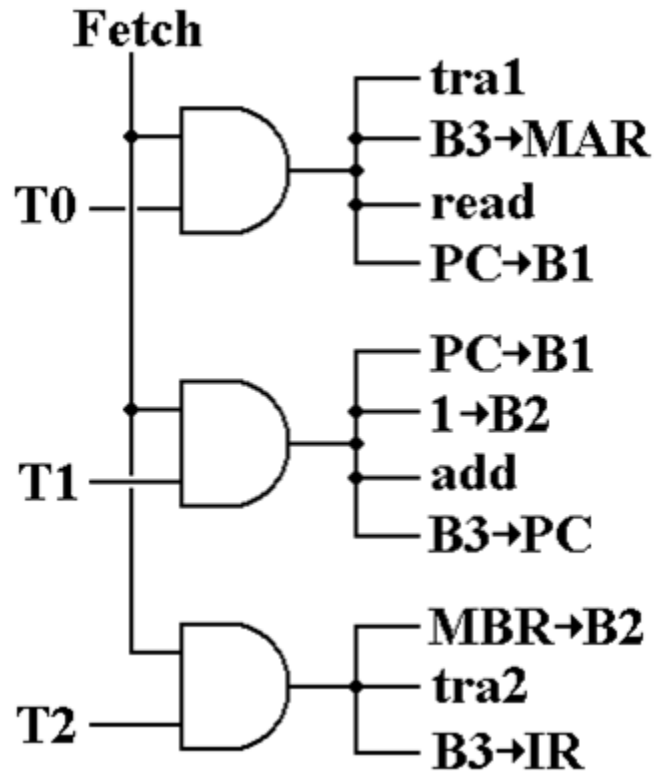
Fetch, T1: (PC) \rightarrow B1, 4 \rightarrow B2, **add**, B3 \rightarrow PC.

Fetch, T2: (MBR) \rightarrow B2, **tra2**, B3 \rightarrow IR.

The Boz-5 Control Signals

- PC → B1 Copy the contents of the PC onto bus B1
- +4 → B2 Copy the constant +4 onto B2.
- MBR → B2 Copy the contents of the MBR onto B2
- **tra1** Causes the ALU to copy the contents of B1 onto B3
- **tra2** Causes the ALU to copy the contents of B2 onto B3
- **add** Causes the ALU to add the contents of B1 and B2, placing the sum onto B3.
- **read** Causes the memory to be read; place the results in MBR
- Bus3 → MAR Copy the contents of B3 to the MAR
- Bus3 → PC Copy the contents of B3 to the PC
- Bus3 → IR Copy the contents of B3 to the IR

Hardwired Signal Generation



- The first phase of the fetch sequence has Fetch = 1 and T0 = 1.
- If Fetch = 1 and T0 = 1 then
tra1 = 1 (it is asserted)
B3 → MAR = 1
read = 1
PC → B1 = 1

Microprogrammed Signals

	PC → Bus1	+1 → Bus2	MBR → Bus2	Bus3 → MAR	Bus3 → PC	Bus3 → IR	add	tra1	tra2	read
T0	1	0	0	1	0	0	0	1	0	1
T1	1	1	0	0	1	0	1	0	0	0
T2	0	0	1	0	0	1	0	0	1	0

- The microprogram can be written as

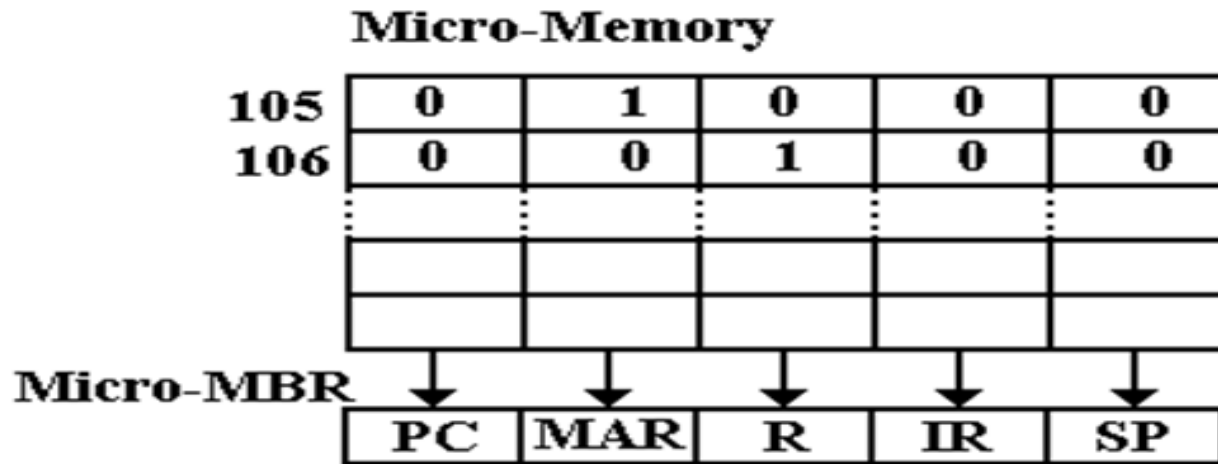
10 0100 0101 0x245

11 0010 1000 0x328

00 1001 0010 0x092

Microprogramming Example

- Consider the micro-memory associated with bus B1.



- At address 105 we have MAR \rightarrow B1.
- At address 106 we have R \rightarrow B1.

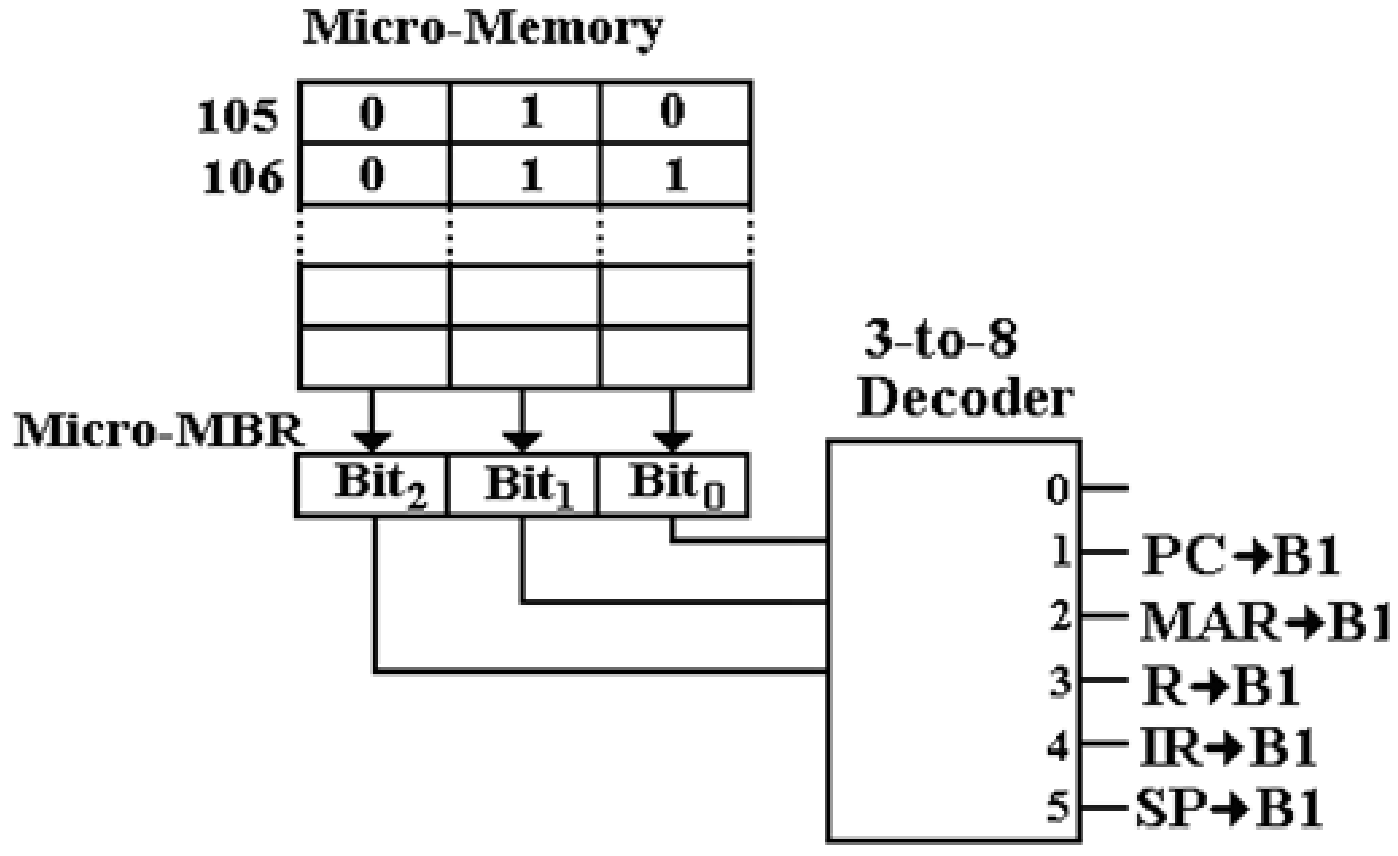
Horizontal and Vertical Microcode

- Consider a bus, B1, that can be fed by seven different signal sources.
- In **horizontal microcode**, each signal has a bit in the micro-memory. The B1 field would have 8 bits.
- In **vertical microcode**, the field would have a binary encoding to indicate the single source to be placed on the bus; here 3 bits.

Advantages of Vertical Microcode

- One advantage is that it allows a “narrower micro–memory”, fewer bits per word in the micro–memory. But memory is cheap.
- The major advantage is that it prevents the assertion of two or more data sources on a given bus or two or more simultaneous ALU operations.

B1 With Vertical Microcode



Typical Microinstruction Format

- Each microinstruction includes the address of the instruction to be executed next.
- Here is a format that supports a branch. There are two 8-bit addresses. The signal S2 will indicate whether or not the branch is taken.
- Non-branching instructions have the same value in both fields.

Micro-Op	B1	B2	B3	ALU	M1	M2	S2 = 0	S2 = 1
4 bits	4 bits	4 bits	4 bits	4 bits	4 bits	4 bits	8 bits	8 bits

Why Store Addresses in the Microcode

- Here are two options for a sequence of control signals, taken out of context. The first uses a conditional branch.
- 0x02C IR → B1, R → B2, **add**, B3 → MAR
0x02D If D = 0 Go To 0x030
0x02E READ
- Similar code in the modern style
0x02C IR → B1, R → B2, **add**, B3 → MAR, 0x030, 0x02E

Maurice Wilkes

- Maurice Wilkes worked in the Computing Laboratory at Cambridge University beginning in 1936, but mostly from 1945 as he served in WW 2.
- On May 6, 1949 the EDSAC was first operational, computing the values of N^2 for $1 \leq N \leq 99$. In 1951, Wilkes published The Preparation of Programs for Electronic Digital Computers, the first book on programming.
- Also in 1951, Wilkes published a paper “The Best Way to Design an Automatic Calculating Machine” that described a technique that he called microprogramming. This technique is still in use today and still has the same name.

Wilkes' Motivation

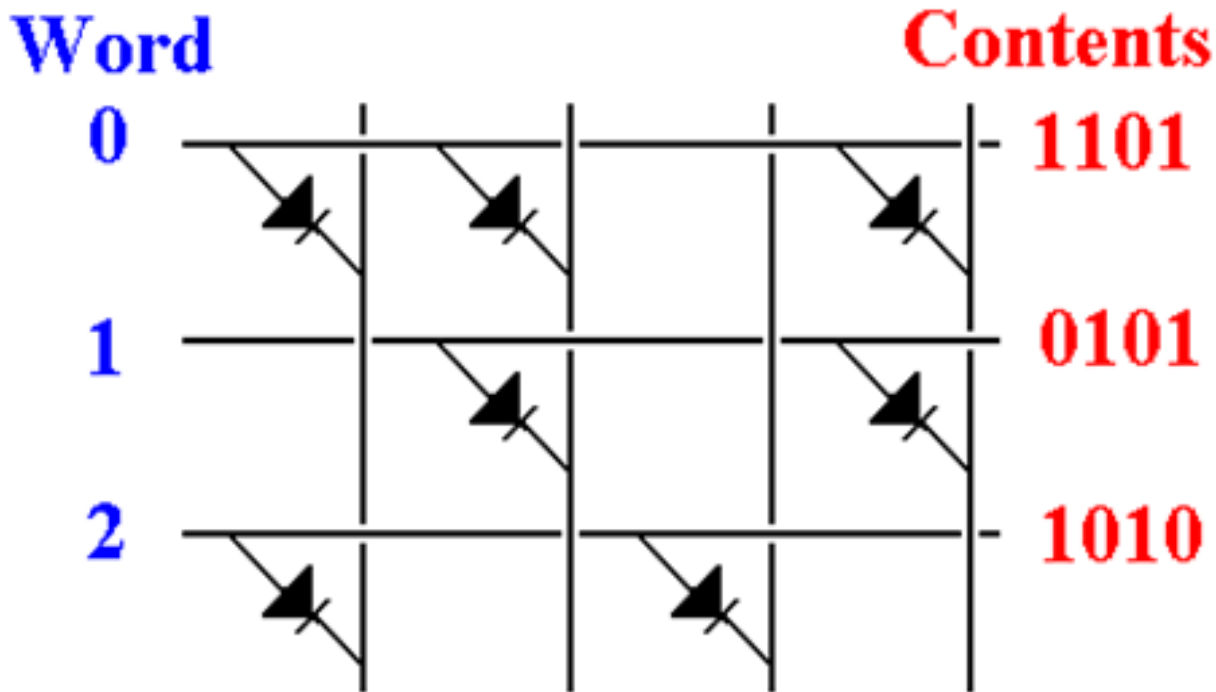
- Here is a direct quote from Maurice Wilkes.
- “As soon as we started programming, we found to our surprise that it wasn't as easy to get programs right as we had thought. ... I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs.”

The Complexity of the Control Unit

- After his visit to the United States, Wilkes started to worry about the complexity of the control unit of the EDVAC, then in design. Here is what he wrote later.
- *“It was not, I think, until I got back to Cambridge that I realized that the solution was to turn the control unit into a computer in miniature by adding a second matrix to determine the flow of control at the micro-level and by providing for conditional microinstructions”.*

A Diode Matrix

- A diode memory is just a collection of diodes connected in a matrix.



Problems in the 1950's

- In 1958, the EDSAC 2 became operational; it was the first microprogrammed computer. The control unit used ROM made from magnetic cores.
- There were two reasons that Wilkes' idea did not take off in the 1950's.
 1. The simple instruction sets of the time did not demand microprogramming, and
 2. The methods for fabricating a microprogram control store were not adequate.

Microprogramming is Taken Seriously

- IBM introduced the System/360 in 1964. This caused microprogramming to be taken seriously as an option for designing control units. There were three reasons.
 1. The recent availability of memory units with sufficient reliability and reasonable cost.
 2. The fact that IBM took the technology seriously.
 3. The fact that IBM aggressively pushed the memory technology inside the company to make microprogramming feasible.

IBM's Goals for Microprogramming

- “Microprogramming in the System/360 ... has been used to help design a fixed instruction set capable of reaching across a compatible line of machines in a wide range of performances. ... [It] has, however, made it feasible for the smaller models of System/360 to provide the same comprehensive instruction set as the large models.”

IBM's Marketing Problem

- The System/360 was a bold move to replace a number of incompatible designs: the IBM 1401, the IBM 7040, and IBM 7094.
- Each of the older designs had a large customer base with considerable developed code.
- None of the old code would run on the 360.
- Honeywell, along with other companies, began marketing newer computers that would run the old code.
- Customers might abandon IBM.

Emulation on the S/360

- IBM was “spared mass defection of former customers” when engineers working on the Model 30 suggested the use of an extra control store on the micro-programmed control unit to allow the Model 30 to execute IBM 1401 instructions in native mode.
- The effort lead to the ability to execute native mode software for both the IBM 1401 and IBM 700 series. Moss termed their work as “**emulation**”.
- The emulators they designed worked well enough so that many customers never converted legacy software and instead ran it for many years on System/360 hardware using emulation. This was a great marketing success for IBM.

The 1960's and 1970's

- In the 1960s and 1970s, microprogramming was one of the most important techniques used in implementing machines. Through most of that period, machines were implemented with discrete components or MSI (medium-scale integration—fewer than 1000 gates per chip)
- The hardwired implementations were faster, but too costly for most machines. Furthermore, it was very difficult to get the control correct, and changing ROMs was easier than replacing a random logic control unit.
- Eventually, microprogrammed control was implemented in RAM, to allow changes late in the design cycle, and even in the field after a machine shipped.

Benefits of Microprogramming

- As noted above, the primary impediment to adoption of microprogramming was that sufficiently fast control memory was not readily available.
- When the necessary memory became available, microprogramming became popular.
- The main advantage of microprogramming was that it handled difficulties associated with virtual memory, especially restarting instructions after page faults.
- The IBM System 370 Model 138 implemented virtual memory entirely in microcode.

The IBM XT/370

- IBM was attempting to regain dominance in the desktop market. They noted that both the S/370 and the Motorola 68000 used sixteen 32-bit general purpose registers.
- In 1984 IBM announced the XT/370, a “370 on a desktop”. The design used a pair of Motorola 68000s re-microprogrammed to emulate the S/370 instruction set. Two units were required because the control store on the Motorola was too small for the S/370.
- As a computer the project was successful. It failed because IBM wanted full S/370 prices for the software to run on the XT/370.

Side-Effects of Microprogramming

- It is a simple fact that the introduction of microprogramming allowed the development of Instruction Set Architectures of almost arbitrary complexity.
- The VAX series, marketed by the Digital Equipment Corporation, is usually seen as the “high water mark” of microprogrammed designs. The later VAX designs supported an Instruction Set Architecture with more than 300 instructions and more than a dozen addressing modes.

Microprogramming and Memory Technologies

- The drawback of microcode has always been memory performance; the CPU clock cycle is limited by the time to read the memory.
- In the 1950's, microprogramming was impractical for two reasons.
 1. The memory available was not reliable, and
 2. The memory available was the same slow core memory as used in the main memory of the computer.

Microprogramming and Memory Technologies

- In the late 1960's, semiconductor memory (SRAM) became available for the control store. It was ten times faster than the DRAM used in main memory. This speed difference that opened the way for microcode.
- In the late 1970's, cache memories using the SRAM became popular. At this point, the CROM lost its speed advantage.
- For these reasons, instruction sets invented since 1985 have not relied on microcode.

Microprogramming: The Late Evolution

- Events that lead to the reduced emphasis on microprogramming include:
 1. The availability of VLSI technology, which allowed a number of improvements, including on-chip cache memory, at reasonable cost.
 2. The availability of ASIC (Application Specific Integrated Circuits) and FPGA (Field Programmable Gate Arrays), each of which could be used to create custom circuits that were easily tested and reconfigured.
 3. The beginning of the RISC (Reduced Instruction Set Computer) movement, with its realization that complex instruction sets were not required.

What Happened to Microcode?

- Control units for RISC designs tend not to use microprogramming, but the simpler and faster hardwired designs.
- One reason is that the simplicity of the control unit does not require microprogramming.
- Another possible reason is that the speed of a modern pipelined control unit requires control signals to be issued at a rate faster than SRAM read-only memory can deliver them.