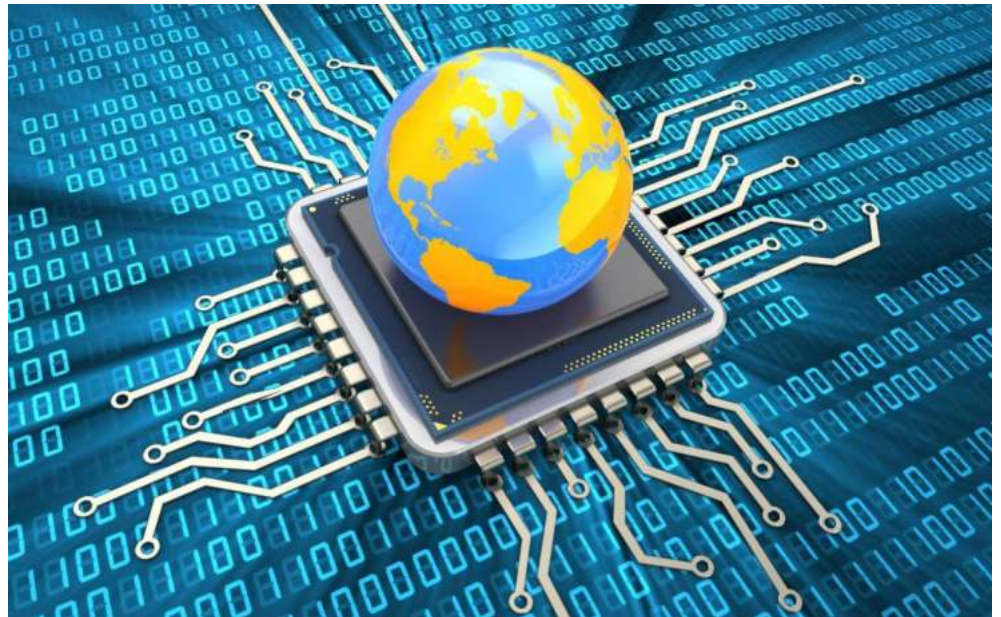


A DISCUSSION ON ECODING: HARDWIRED VS. MICROPROGRAMMED CONTROL

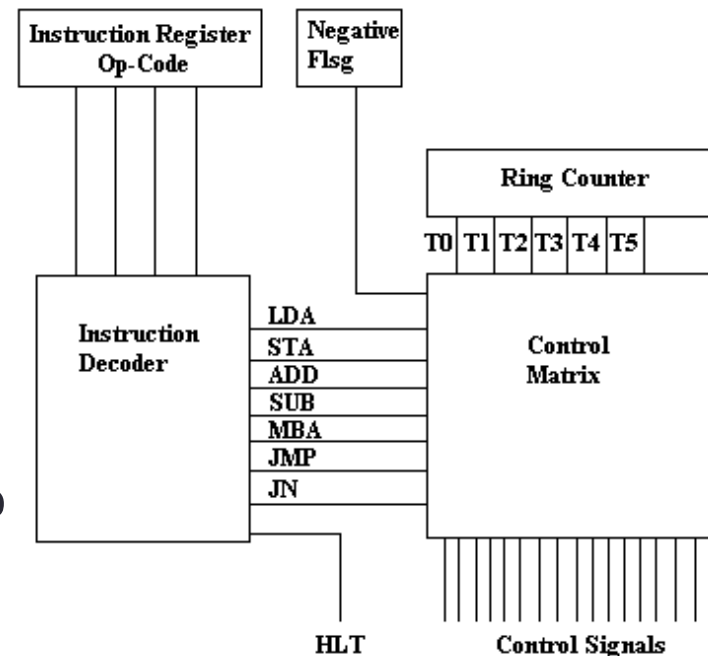


How does the control unit really function?

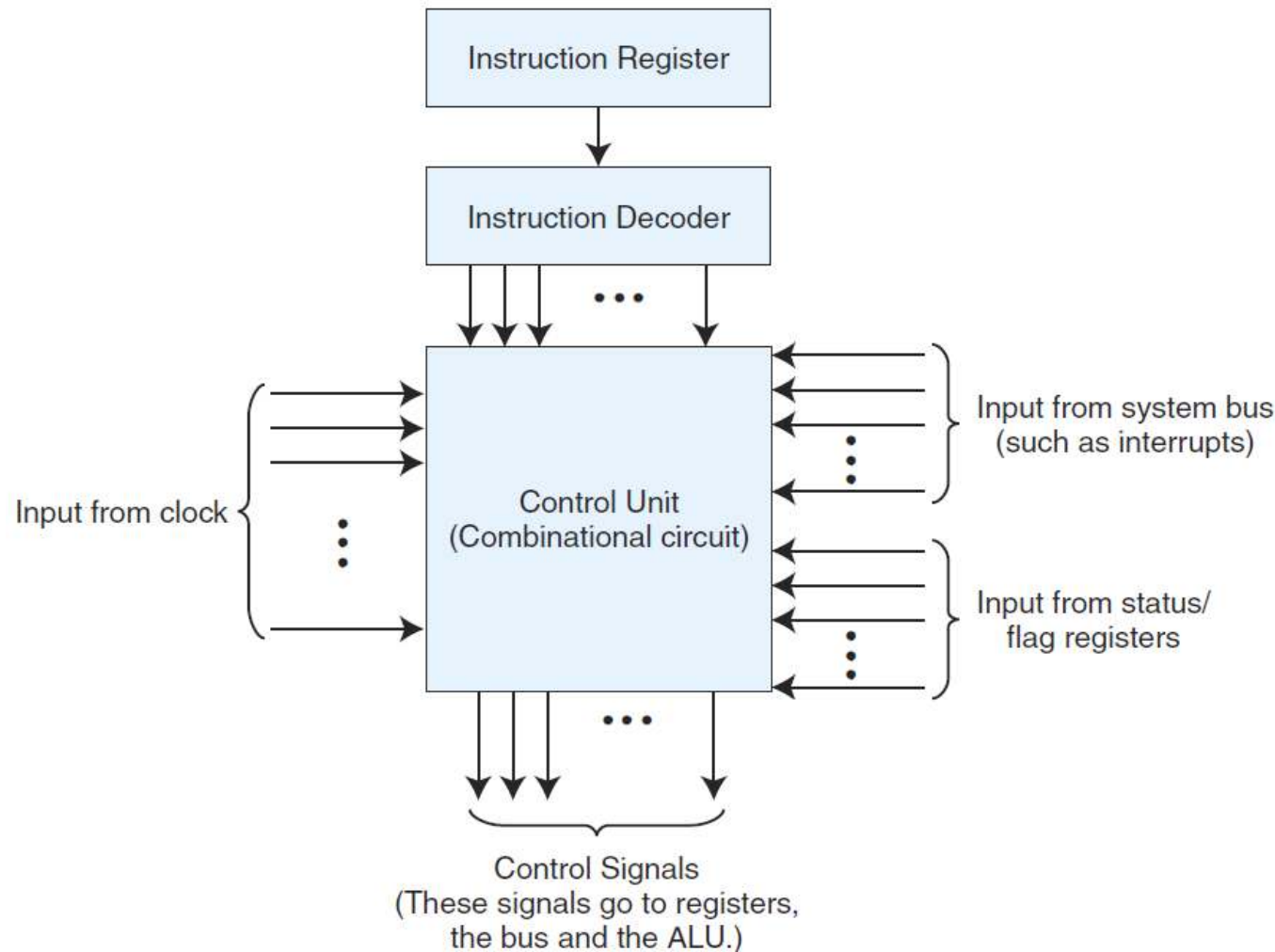
- On instruction execution, CPU submits control signals to components to make things happen.
- **For Example** → "add" instruction takes the data in MBR and add it to the AC and store the result in AC.
- **How do these control lines actually become asserted?**
- **Answer:** there are two ways:
 - ***Hardwired Control***
 - Using sequential circuits to generate signals.
 - ***Microprogrammed Control***
 - Using microinstructions in the control memory to generate control signals.

Hardwired Control

- It is a sequential circuit that generates control signals → NAND gates, flip flops, counters...etc
- The circuit needs to be special → can use opcode bits, flag bits, bus signal, clock signal → to generate control signals to drive different components in the computer
- It is physically connects all of the control lines to the actual machine instructions.
- In MARIE, 4-to-16 decoder could be used to decode the opcode.



Hardwired Control Unit



Hardwired Control

- **Advantage:** It is very fast
- **Disadvantage:** instruction set and control logic are directly tied together by special circuits that are complex and difficult to design or modify.
- **Example** → if someone designs a hardware computer and later decide to extend the instruction set → the physical component in computer must be changed.

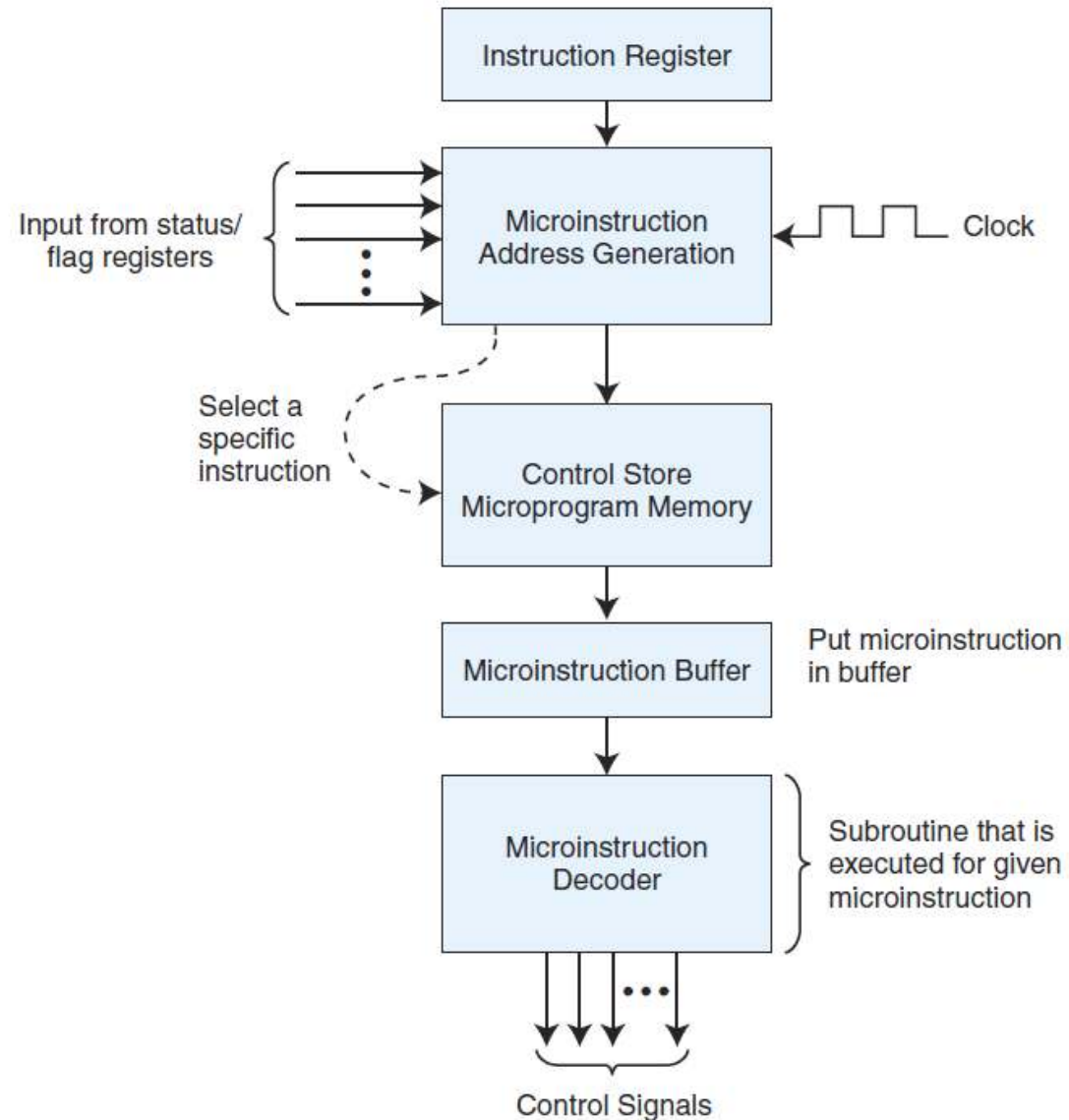
Microprogrammed Control Unit

- Uses software to generate the control signals.
- It converts the instruction into the appropriate control signals.
- The microprogram is essentially an interpreter, written in *microcode*, that is stored in *ROM* which is often referred to as the *control store*.
- Microprogramming is flexible, simple in design, and lends itself to very powerful instruction sets.
- for example, if your machine has no multiplication statement, it can be implemented in the microcode.

Microprogrammed Control Unit

- **Advantage:** flexible, simple in design, and lends itself to very powerful instruction sets
- **Disadvantage:**
 - All instructions must go through additional level of interpretation
→ slowing down the program execution → time cost
 - Additional cost associated with the actual development

Microprogrammed Control Unit



Hardwired Vs. Microprogrammed Control Units

	HARDWIRED Control Unit	MICROPROGRAMMED Control Unit
Way of produceing signals	combinational circuits	microinstruction in the control memory
Speed	fast	slow (needs more memory access)
Modification	redesign the unit (hardware changing)	just update the memory with new set of microinstruction
Cost	more	less
Handling complex instructions	difficult	easier
Instruction Decoding	difficult	easier
Instruction Set	small	large
Require Control Memory	No	Yes
Applied in	Processors that use a simple instruction set known as the Reduced Instruction Set Computers (RISC)	Processors based on complex instruction set known as the Complex Instruction Set Computers (CISC)

Timing in Control Unit (both approaches)

- The control unit is responsible for the actual timing signals that direct all data transfers and actions. These signals are generated in sequence with a simple binary counter called *Ring Counter*.
- *For example* → the timing signals for MARIE is T1, T2, T3, T4, T5, T6, T7, and T8. These signals control when actions can occur.
- A fetch for an instruction might occur only when T1 is activated, whereas the fetch for an operand may occur only when T4 is activated.

REAL WORLD EXAMPLES OF COMPUTER RCHITECTURES



MIPS

Microprocessor without Interlocked Pipeline Stages

Introduction

- MARIE's performance could be significantly improved if by adding more registers. Making things easier for the programmer is a different matter.
- **For example:** "Add M[32], 4" for read the value in location 32 in memory, increment it by 4 and then store the result in AC.

CISC and RISC

- If an instruction uses memory in the execute cycle → slow process
- We need to redesign the instruction set → so few instructions can use the memory in the execute cycle
- We reduce the memory access to improve instructions → because memory is slower
- CISC architecture was created → as initial architecture focuses on memory access.
- CISC = Complex Instruction Set Computer
- We have a simple fetch cycle and 100s of execute cycles

CISC and RISC

- CISC machines have a large number of instructions, with complex layouts. Many of these instructions are quite complicated, performing multiple operations when a single instruction is executed
 - (e.g., it is possible to do loops using a single assembly language instruction).
- The basic problem with CISC machines is that a small subset of complex CISC instructions slows the systems down considerably.
- It would be the compiler's responsibility to produce efficient code for the ISA.

RISC

- Machines utilizing the philosophy of reducing codes are called RISC machines.
- The main objective of RISC machines is to simplify instructions so they can execute more quickly.
 - **Each instruction performs only one operation,**
 - **they are all the same size,**
 - **they have only a few different layouts,**
 - **all arithmetic operations must be performed between registers (data in memory cannot be used as operands).**
- Virtually all new instruction sets (for any architectures) since 1982 have been RISC, or some sort of combination of CISC and RISC.

CISC and RISC

- The x86 family of Intel architectures is known as a **CISC** (**C**omplex **I**nstruction **S**et **C**omputer) machine.
- Pentium family and the MIPS architectures are examples of **RISC** (**R**educed **I**nstruction **S**et **C**omputer) machines

Intel Architectures

- Intel's first popular chip, the 8086 produced in 1979.
 - 16-bit data
 - 20-bit addresses (addressing up to $2^{20} = 1$ MB).
 - It has 4 general purpose registers named AX, BX, CX and DX
 - **Example of other registers:**
 - IP (Instruction Pointer): like AC in MARIE.
 - SI (Source Index): used as source pointer of string operations.
 - DI (Destination Index): used as a destination pointer for string operations.
 - Status Flag Registers: for carry, overflow, interrupt.

Intel Architectures

- In 1980, the 8087 is introduced, which added floating-point instructions.
- Many new chips were introduced that used essentially the same ISA as the 8086,
 - 80286 in 1982
 - 80386 in 1985
 - 80386 in 1985 was a 32-bit word size (data and address busses)
 - 80486 in 1989, added a high-speed **cache** memory.
 - 80486 DX2 and 80486 DX4, were another new versions of 80486.

Intel Architectures

- A new series is introduced called “Pentium” because it was unable to trademark the numbers)
 - ✓ Pentium in 1993, 32-bit registers and a 64-bit data bus.
 - **superscalar** design (multiple ALUs, so it may execute multiple instruction at a time.
 - ✓ Pentium Pro in 1995,
 - ✓ Pentium II in 1997, added MMX technology to deal with multimedia.
 - ✓ Pentium III 1999, added increased support for 3D graphics.
 - ✓ *Pentium 4*, 1.4GHz clock, hyper-pipeline
 - implements “Netburst” microarchitecture (The processors in the Pentium family, up to this point, had all been based on the same microarchitecture)

Intel was conforming to the current trend by moving away from CISC and toward RISC

Intel Architectures

- Itanium marked Intel's first 64-bit chip (IA-64).
 - Itanium in 2001, 128-bit for integer and floating-point
 - address up to 16 GB
 - Itanium 2 in 2002
- Core i7 in 2008, 731 million transistors
- Core i7 in 2010, 1 Billion transistors
- Xeon in 2012, 2 billion transistors

MIPS Architectures

- MIPS family of CPUs has been successful and flexible.
- The MIPS R3000, R4000, R5000, R8000, and R10000 are some of the many registered trademarks.
- MIPS chips are used in embedded systems, in addition to
 - **computers (such as Silicon Graphics machines)**
 - **computerized toys (Nintendo and Sony)**
 - **Cisco uses MIPS CPUs as well.**
- MIPS32 (32-bit) and MIPS64 (64-bit)