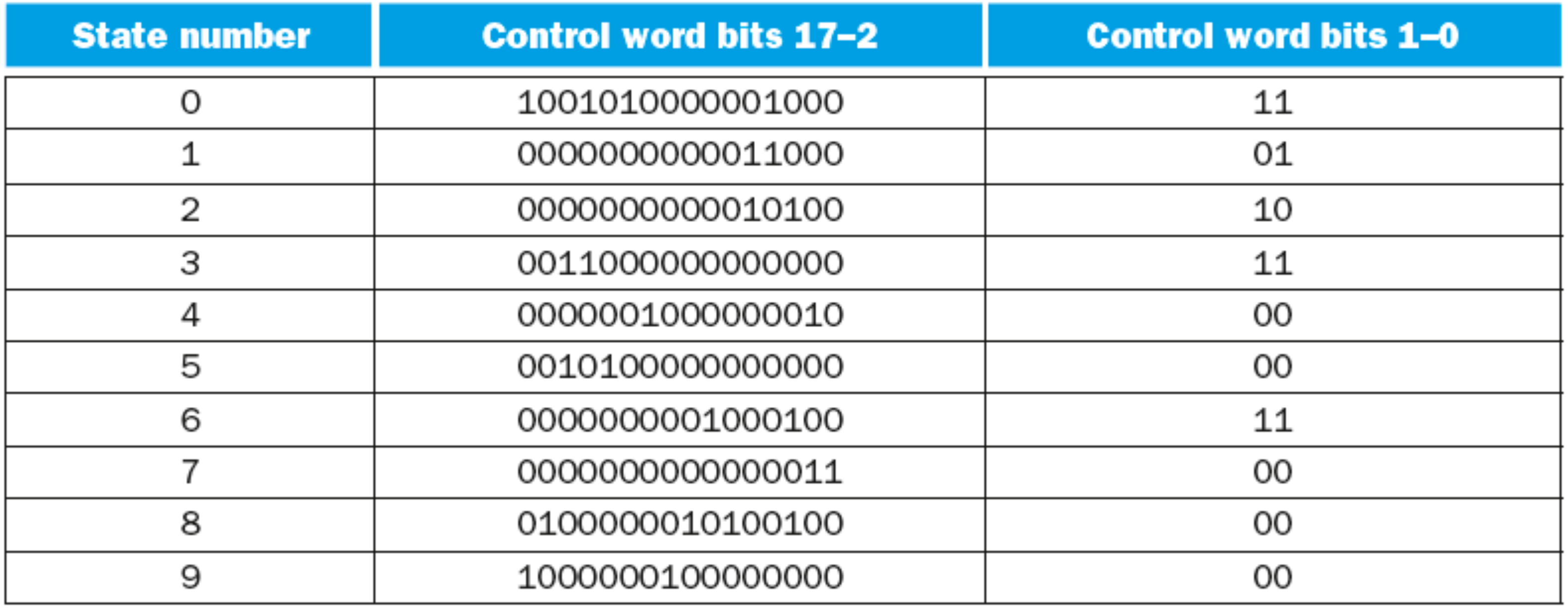
Control ROM Contents

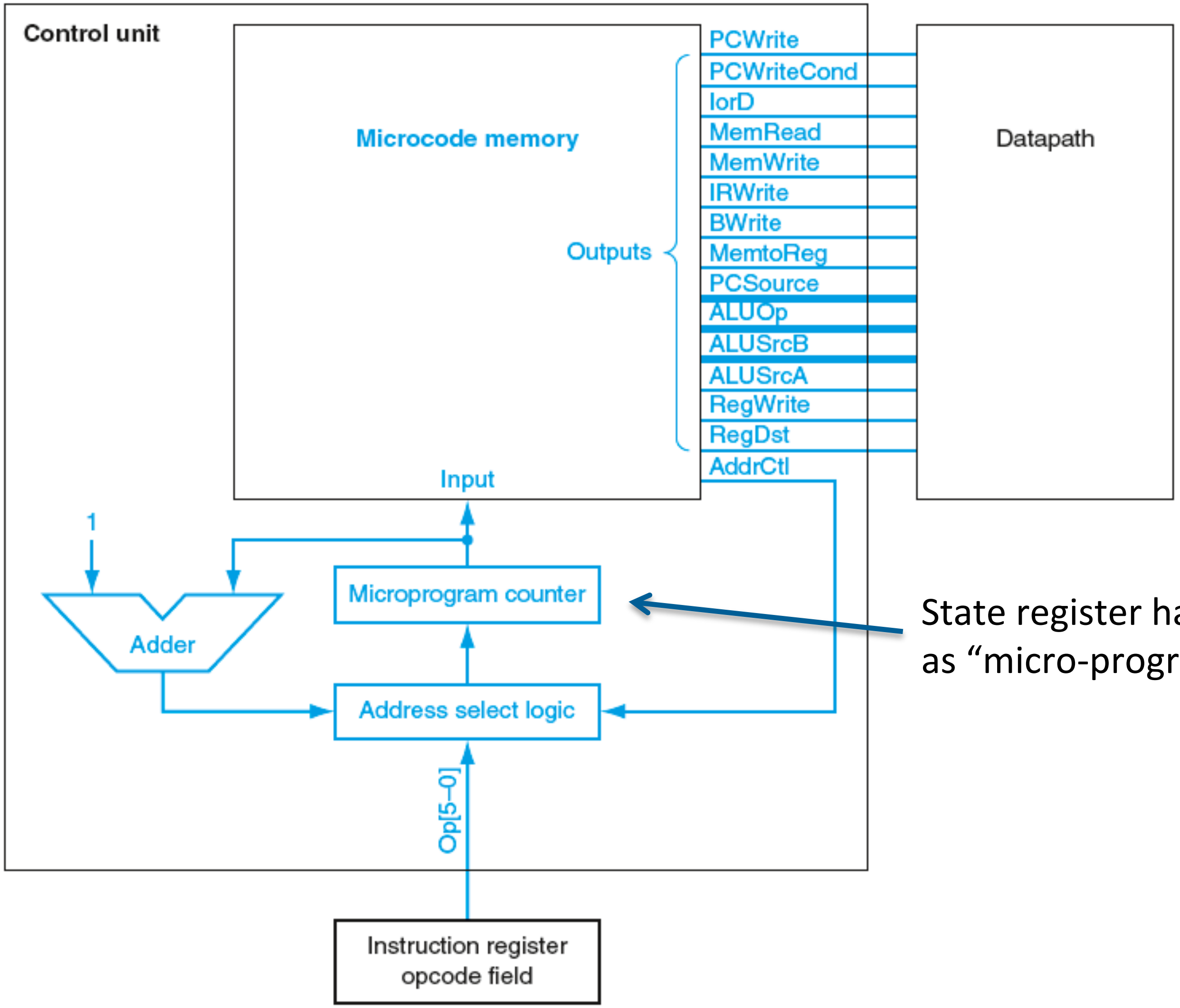
Micro-instructions are contained in the control store (ROM)



State number	Control word bits 17–2	Control word bits 1–0
0	1001010000001000	11
1	0000000000011000	01
2	0000000000010100	10
3	0011000000000000	11
4	0000001000000010	00
5	0010100000000000	00
6	0000000001000100	11
7	0000000000000011	00
8	0100000010100100	00
9	1000000100000000	00

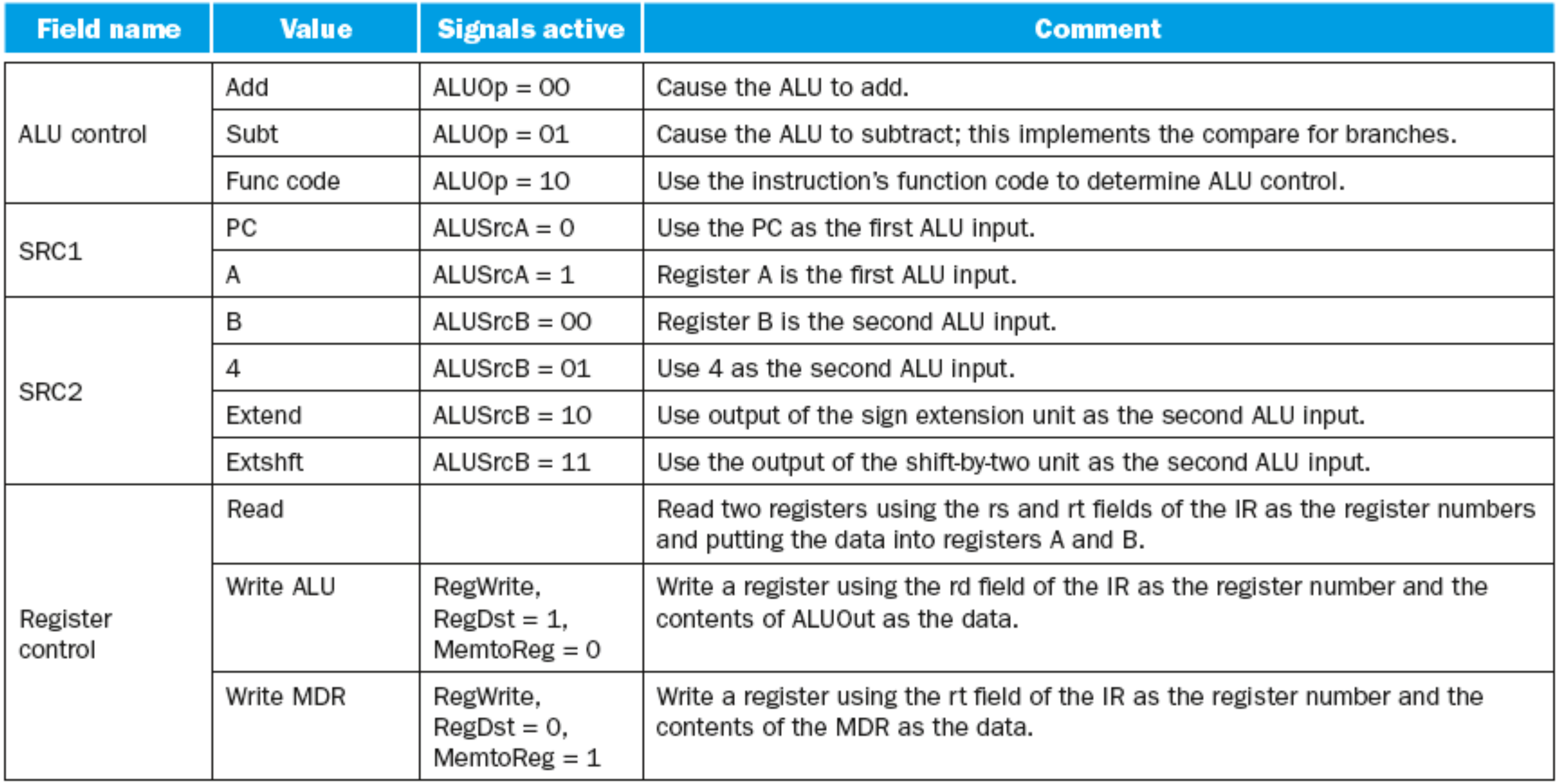
Each micro-instruction takes one clock cycle

Micro-programming



State register has been renamed as “micro-program counter”

Micro-Instruction Fields

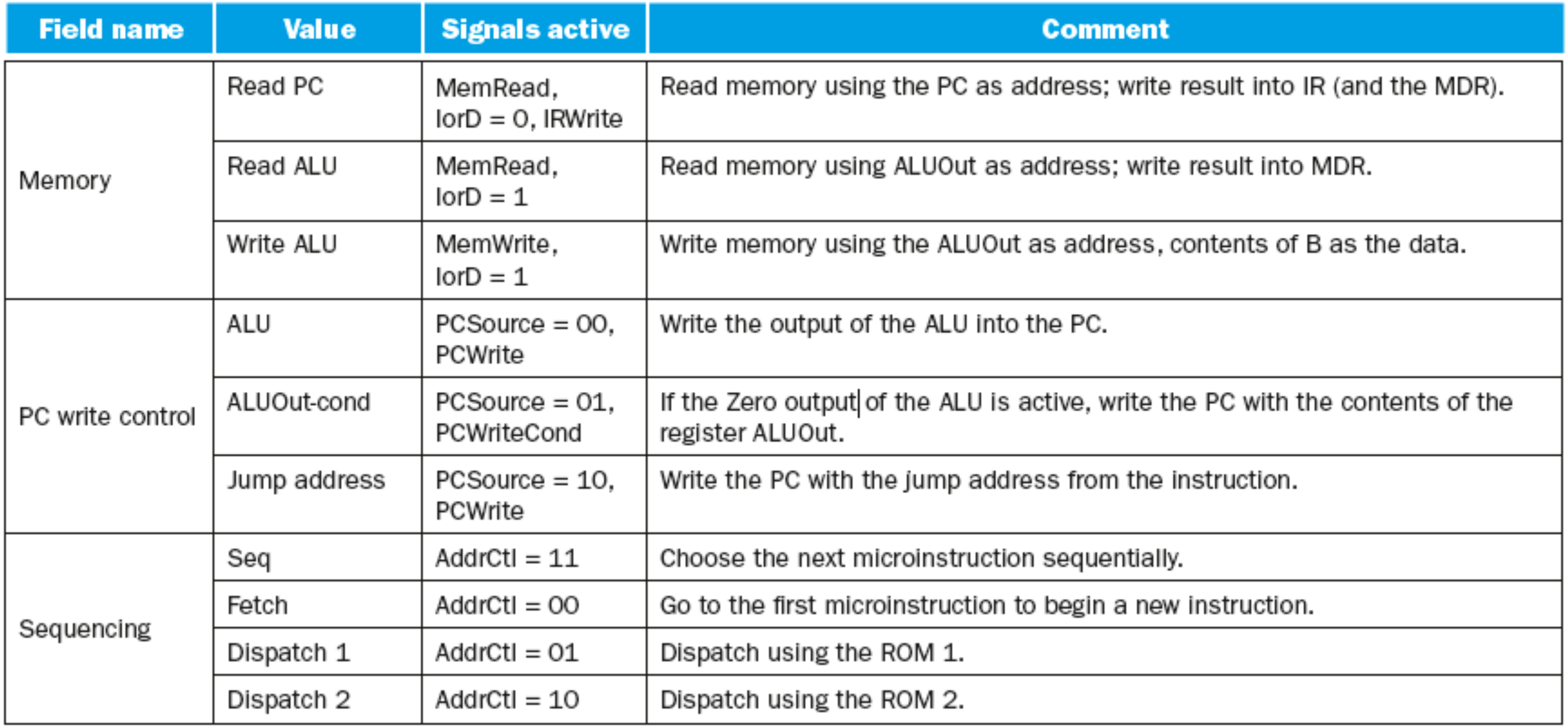


Field name	Value	Signals active	Comment
ALU control	Add	ALUOp = 00	Cause the ALU to add.
	Subt	ALUOp = 01	Cause the ALU to subtract; this implements the compare for branches.
	Func code	ALUOp = 10	Use the instruction's function code to determine ALU control.
SRC1	PC	ALUSrcA = 0	Use the PC as the first ALU input.
	A	ALUSrcA = 1	Register A is the first ALU input.
SRC2	B	ALUSrcB = 00	Register B is the second ALU input.
	4	ALUSrcB = 01	Use 4 as the second ALU input.
	Extend	ALUSrcB = 10	Use output of the sign extension unit as the second ALU input.
	Extshft	ALUSrcB = 11	Use the output of the shift-by-two unit as the second ALU input.
Register control	Read		Read two registers using the rs and rt fields of the IR as the register numbers and putting the data into registers A and B.

	Write ALU	RegWrite,
		RegDst = 1,
		MemtoReg = 0
			Write a register using the rd field of the IR as the register number and the contents of ALUOut as the data.

	Write MDR	RegWrite,
		RegDst = 0,
		MemtoReg = 1
			Write a register using the rt field of the IR as the register number and the contents of the MDR as the data.

Micro-Instruction Fields



Field name	Value	Signals active	Comment
Memory	Read PC	MemRead, lorD = 0, IRWrite	Read memory using the PC as address; write result into IR (and the MDR).
	Read ALU	MemRead, lorD = 1	Read memory using ALUOut as address; write result into MDR.
	Write ALU	MemWrite, lorD = 1	Write memory using the ALUOut as address, contents of B as the data.
PC write control	ALU	PCSource = 00, PCWrite	Write the output of the ALU into the PC.
	ALUOut-cond	PCSource = 01, PCWriteCond	If the Zero output of the ALU is active, write the PC with the contents of the register ALUOut.
	Jump address	PCSource = 10, PCWrite	Write the PC with the jump address from the instruction.
Sequencing	Seq	AddrCtl = 11	Choose the next microinstruction sequentially.
	Fetch	AddrCtl = 00	Go to the first microinstruction to begin a new instruction.
	Dispatch 1	AddrCtl = 01	Dispatch using the ROM 1.
	Dispatch 2	AddrCtl = 10	Dispatch using the ROM 2.

Micro-programming

- Each micro-instruction performs one step in a machine instruction
- Each micro-instruction takes one clock cycle
- A machine instruction corresponds to a “micro-program”
- A micro-program is a sequence of micro-instructions
- Control memory containing micro-programs is on CPU chip
- Extra time is needed to retrieve and execute micro-programs

Micro-programming

- Time required to retrieve micro-progams slows the system
- Micro-programming provides flexibility
- Complex instructions are easier to implement as micro-code
- CISC systems employ micro-programming
- Changing micro-code changes behavior of the system
- RISC systems use faster hardwired logic instead

Micro-code Encoding

- Each micro-instruction in our conrol ROM is 18 bits wide
- None of the fields within the micro-instructions are encoded
- This is fast since no decoding is required
- Systems of this type are said to be “minimally encoded”
- This is also called “horizonal micro-code”
- Micro-code is sometimes called firmware
- Firmware is harder to change than software
- Hardware is more difficult to change that firmware

Micro-code Encoding

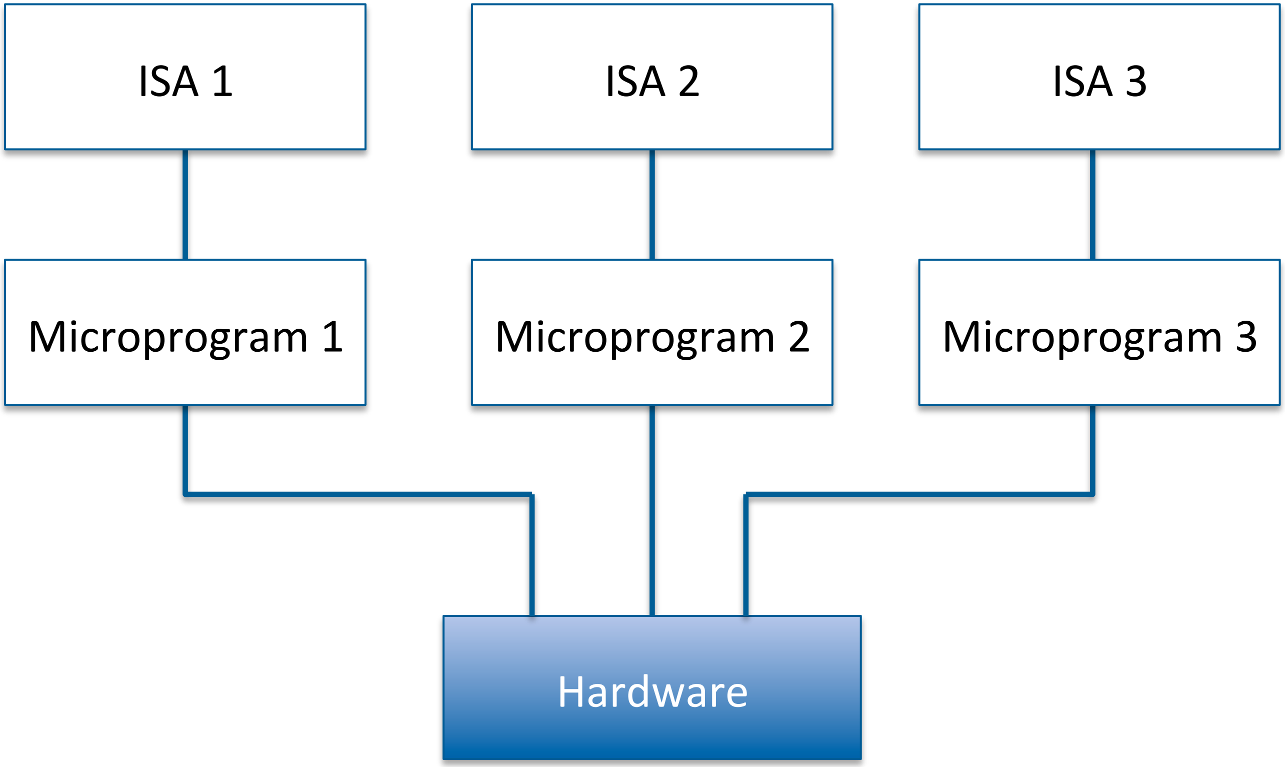
- A writeable control store employs RAM for micro-code
- More realistic systems are more complex than our core MIPS
- Such systems could require very wide micro-instructions
- Encoding fields within these micro-instructions would save bits
- However these take longer to process due to need for decoding
- These systems employ maximally encoding (“vertical micro-code)

Emulation

- Micro-programming enables one system to behave like another
- This is called “emulation”
- It allows the same hardware to be used with different ISAs
- ISA is the instruction set architecture
- Different machine code can execute on the same hardware
- The micro-code is said to interpret the machine instructions

Emulation

Emulation allows the hardware to run different machine programs


ISA 1
ISA 2
ISA 3
Microprogram 1
Microprogram 2
Microprogram 3
Hardware