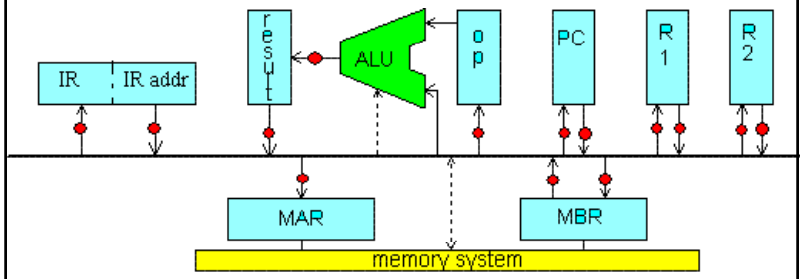
More on Microcode

COMP375 Computer Architecture and Organization

Fetch Execute Cycle

1. Fetch the instruction from the memory address in the Program Counter register
2. Increment the Program Counter
3. Decode the type of instruction
4. Fetch the operands
5. Execute the instruction
6. Store the results

Simple CPU



Instruction Fetch

1. Fetch the instruction from the memory address in the Program Counter register
- Copy the program counter to the Memory Address Register
- Tell the memory system to read.
- Wait for the read to complete
- Copy from the Memory Data Register to the Instruction Register

Instruction Fetch

bus → IR	IR adr → bus	result → bus	ALU → result	bus → oprnd	bus → PC	PC → bus	bus → R1	R1 → bus	bus → R2	R2 → bus	bus → MAR	bus → MBR	MBR → bus	ALU fun	Mem func
						X					X				read
															wait
X													X		

2. Increment the Program Counter

- Copy the Program Counter to the ALU input register.
- Set the ALU function to increment.
- Copy the ALU output register to the Program Counter

2. Increment the Program Counter

bus → IR	IR adr → bus	result → bus	ALU → result	bus → oprnd	bus → PC	PC → bus	bus → R1	R1 → bus	bus → R2	R2 → bus	bus → MAR	bus → MBR	MBR → bus	ALU fun	Mem func
			X			X								inc	
		X			X										

4. Fetch the operands

Assuming direct addressing

- Copy the address portion of the instruction register to the Memory Address Register.
- Tell the memory system to read.
- Wait for the read to complete
- Copy from the Memory Data Register to the appropriate data or ALU Register

4. Fetch the operands

bus → IR	IR adr → bus	result → bus	ALU → result	bus → oprnd	bus → PC	PC → bus	bus → R1	R1 → bus	bus → R2	R2 → bus	bus → MAR	bus → MBR	MBR → bus	ALU func	Mem func
	X										X				read
															wait
							?						X		

5. Execute the instruction

Assume an arithmetic instruction

- Copy the operand from a data register to an ALU input register.
- Set the ALU function according to the opcode field of the instruction register

5. Execute the instruction

bus → IR	IR adr → bus	result → bus	ALU → result	bus → oprnd	bus → PC	PC → bus	bus → R1	R1 → bus	bus → R2	R2 → bus	bus → MAR	bus → MBR	MBR → bus	ALU func	Mem func
				X				X						add	
			X							X					

Assume adding R1 and R2

6. Store the results

- Copy the output ALU register to the appropriate data register.

6. Store the results

bus → IR	IR adr → bus	result → bus	ALU → result	bus → oprnd	bus → PC	PC → bus	bus → R1	R1 → bus	bus → R2	R2 → bus	bus → MAR	bus → MBR	MBR → bus	ALU func	Mem func
		X					X								

Add R1, xyz (memory direct)

bus → IR	IR adr → bus	result → bus	ALU → result	bus → oprnd	bus → PC	PC → bus	bus → R1	R1 → bus	bus → R2	R2 → bus	bus → MAR	bus → MBR	MBR → bus	ALU func	Mem func
						X					X				read
															wait
	X										X				
			X			X								inc	
		X			X										
	X										X				read
				X				X							wait
			X										X	add	
		X					X								

What control lines are set to copy the value from the result reg to R1?

bus → IR	IR adr → bus	result → bus	ALU → result	ALU fun	bus → oprnd	bus → PC	PC → bus	bus → R1	R1 → bus	bus → R2	R2 → bus	bus → MAR	bus → MBR	MBR → bus	Mem func
		A			B			C	E	D					

1. A & E
2. B & C
3. A & D
4. A & C

Layers

- Applications
- Middleware
- High level languages
- Machine Language
- Microcode
- Logic circuits
- Gates
- Transistors
- Silicon structures

Implementing Machine Language

- Each machine language instruction is implemented by a series of microcode steps.
- Each instruction uses common microcode steps to fetch the instruction and increment the program counter.

Example Layers

C++

A = A + B;

Assembler

Load	R1, A
Add	R1, B
Store	R1, A

Load R1, A

Instruction fetch not shown

bus → IR	IR adr → bus	result → bus	ALU → result	bus → opnd	bus → PC	PC → bus	bus → R1	R1 → bus	bus → R2	R2 → bus	bus → MAR	bus → MBR	MBR → bus	ALU fun	Mem func
	X										X				read
															wait
							X						X		

Add R1, B

Instruction fetch not shown

bus → IR	IR adr → bus	result → bus	ALU → result	bus → opnd	bus → PC	PC → bus	bus → R1	R1 → bus	bus → R2	R2 → bus	bus → MAR	bus → MBR	MBR → bus	ALU fun	Mem func
	X										X				read
				X				X							wait
			X										X	add	
		X					X								

Store R1, A

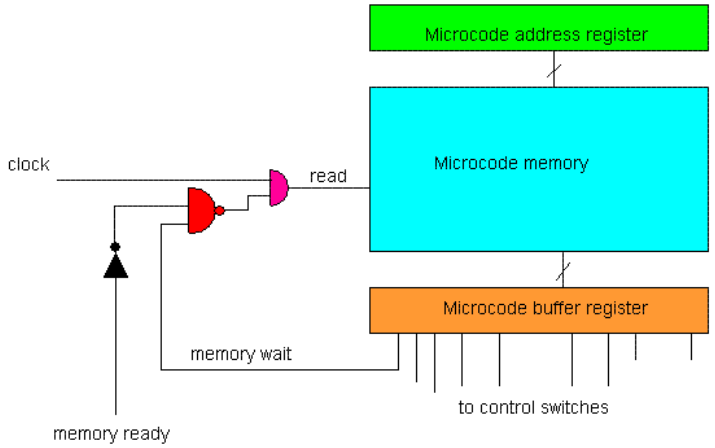
Instruction fetch not shown

bus → IR	IR adr → bus	result → bus	ALU → result	bus → operand	bus → PC	PC → bus	bus → R1	R1 → bus	bus → R2	R2 → bus	bus → MAR	bus → MBR	MBR → bus	ALU fun	Mem func
	X										X				
								X				X			write
															wait

Microcode Steps

- A microcode step is executed each clock cycle.
- For each microcode step, the next line of the microcode store is read.
- The microcode data determines the state of the switches connecting registers to the bus.

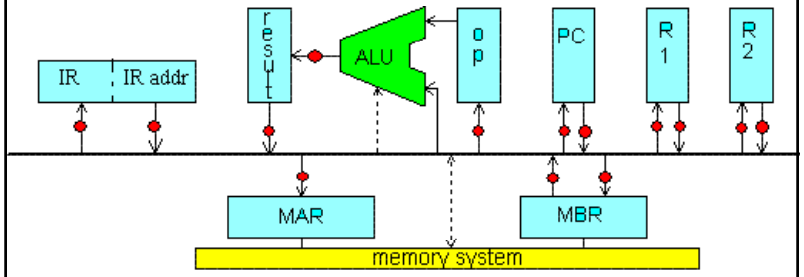
Microcode Store



Where is the Microcode kept?

1. RAM
2. Cache
3. ROM in CPU
4. Registers

Simple CPU



Microcode Programs

- The sequence of switch settings in the microcode store is a microcode program.
- Real microcode programs can conditionally jump to another line in the program.
- Often each line of the microcode contains a field with the address of the next microcode step.

Try It

`add R1, R2`

- Add to R2 to R1 and save the result in R1.

`add R1, R2`

Instruction fetch not shown

bus → IR	IR adr → bus	result → bus	ALU → result	bus → oprnd	bus → PC	PC → bus	bus → R1	R1 → bus	bus → R2	R2 → bus	bus → MAR	bus → MBR	MBR → bus	ALU fun	Mem func
				X				X							
			X							X				add	
		X					X								

Try It

`add R1,[R2]`

- Add to R1 the value pointed to by R2

`add R1, [R2]`

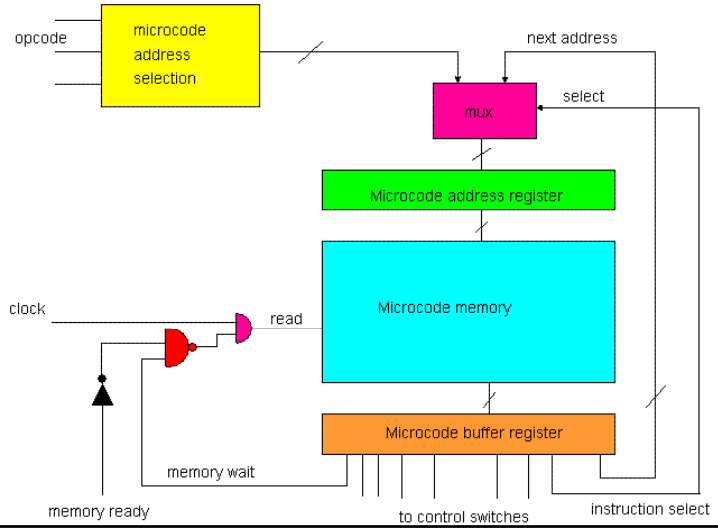
Instruction fetch not shown

bus → IR	IR adr → bus	result → bus	ALU → result	bus → opnd	bus → PC	PC → bus	bus → R1	R1 → bus	bus → R2	R2 → bus	bus → MAR	bus → MBR	MBR → bus	ALU fun	Mem func
										X		X			read
				X				X							wait
			X										X	add	
		X					X								

Opcode Determines Microcode Steps

- After the microcode steps to fetch the instruction and increment the program counter, the next microcode step executed depends on the opcode of the instruction.

Opcode and Microcode Step



Internal CPU Bus

- Only one register can put its value on the bus at a time.
- The value on the bus can be copied into many registers at the same time.

Jump Instruction

Jump to the address given in the instruction

Direct Addressing

bus → IR	IR adr → bus	result → bus	ALU → result	bus → oprnd	bus → PC	PC → bus	bus → R1	R1 → bus	bus → R2	R2 → bus	bus → MAR	bus → MBR	MBR → bus	ALU fun	Mem func
	X				X										

add R1, dog[R2]

Register Indirect with Offset

The operand address is the sum of R2 and the address field of the instruction.

bus → IR	IR adr → bus	result → bus	ALU → result	bus → oprnd	bus → PC	PC → bus	bus → R1	R1 → bus	bus → R2	R2 → bus	bus → MAR	bus → MBR	MBR → bus	ALU fun	Mem func
	X			X											
			X							X				add	
		X									X				read
				X				X							wait
			X										X	add	
		X					X								

Optimizing Microcode

- The more microcode steps required, the longer an instruction will take to execute.
- The value on the bus can be copied into multiple registers
- Instead of copying A to B then B to C, the value in A can be copied to both B and C.
- Unrelated microcode steps can be executed before waiting for a memory access to complete.

Instruction Fetch & Program Counter Increment

bus → IR	IR adr → bus	result → bus	ALU → result	bus → oprnd	bus → PC	PC → bus	bus → R1	R1 → bus	bus → R2	R2 → bus	bus → MAR	bus → MBR	MBR → bus	ALU fun	Mem func
						X					X				read
															wait
X													X		
			X			X								inc	
		X			X										

Instruction Fetch & Program Counter Increment

Combining steps to reduce total time

bus → IR	IR adr → bus	result → bus	ALU → result	bus → oprnd	bus → PC	PC → bus	bus → R1	R1 → bus	bus → R2	R2 → bus	bus → MAR	bus → MBR	MBR → bus	ALU fun	Mem func
			X			X					X			inc	read
		X			X										wait
X													X		