Embedded Systems Design: A Unified Hardware/Software Introduction

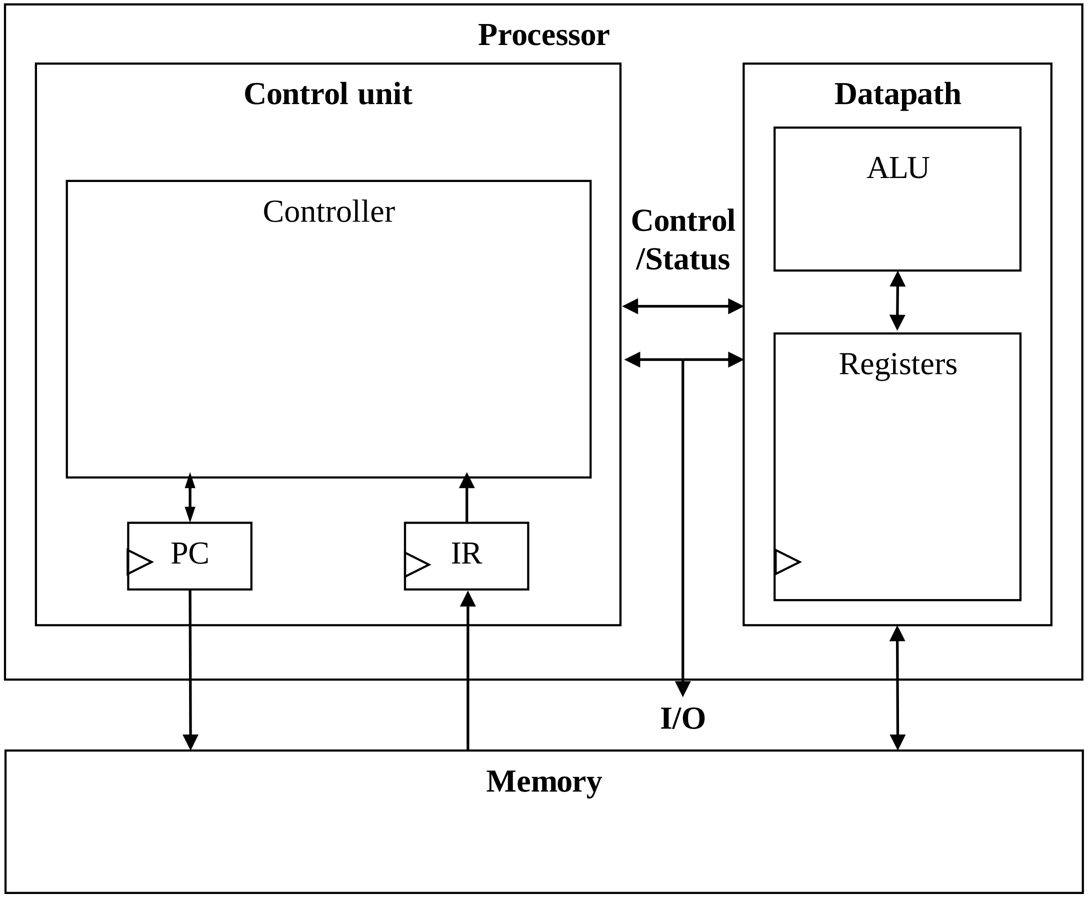
Chapter 3   General-Purpose Processors: Software

Introduction

- General-Purpose Processor
  - Processor designed for a variety of computation tasks
  - Low unit cost, in part because manufacturer spreads NRE over large numbers of units
    - Motorola sold half a billion 68HC05 microcontrollers *in 1996 alone*
  - Carefully designed since higher NRE is acceptable
    - Can yield good performance, size and power
  - Low NRE cost, short time-to-market/prototype, high flexibility
    - User just writes software; no processor design
  - a.k.a. "microprocessor" – "micro" used when they were implemented on one or a few chips rather than entire rooms

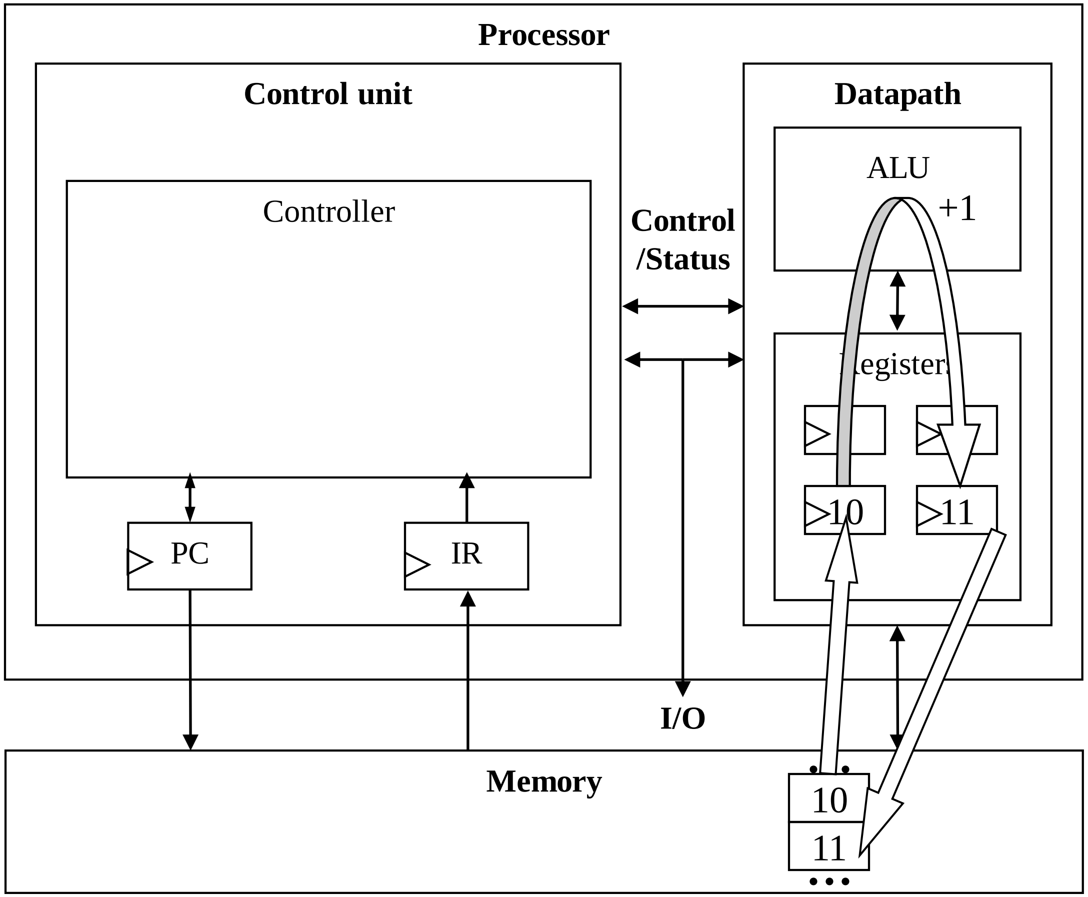
Basic Architecture

- Control unit and datapath
  - Note similarity to single-purpose processor
- Key differences
  - Datapath is general
  - Control unit doesn't store the algorithm – the algorithm is "programmed" into the memory



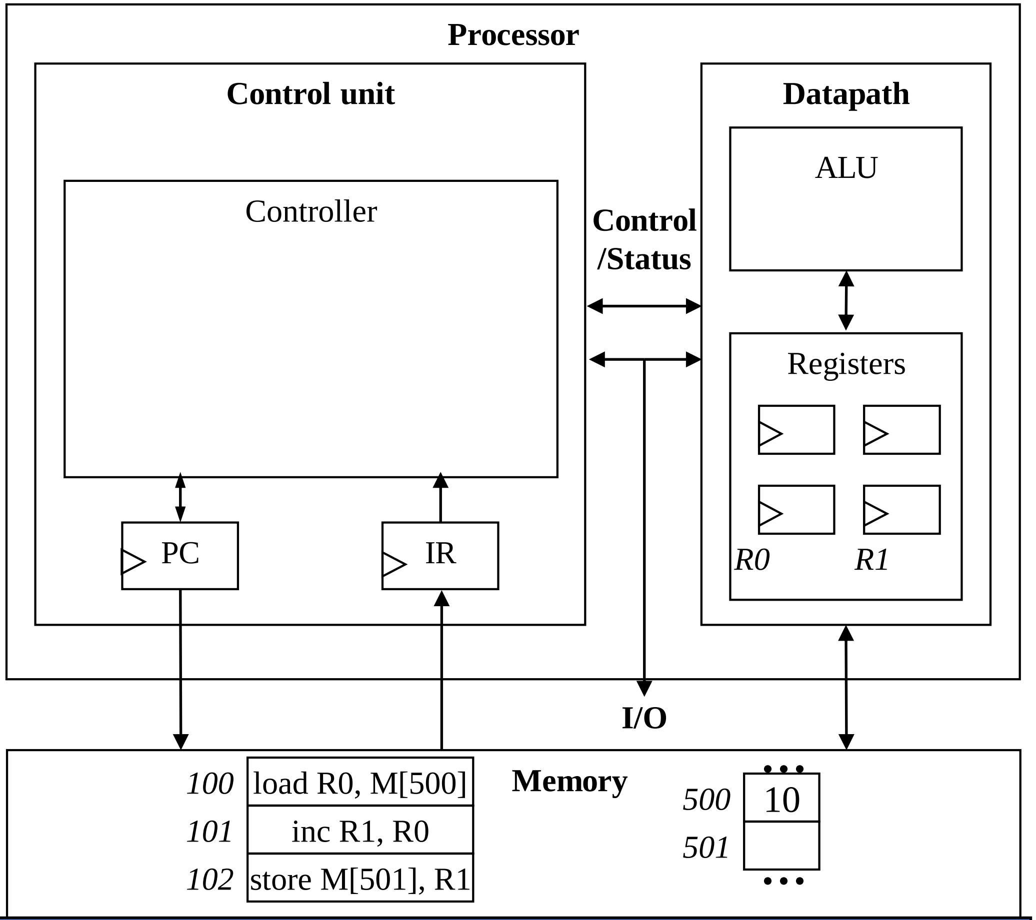
Datapath Operations

- Load
  - Read memory location into register
- ALU operation
  - Input certain registers through ALU, store back in register
- Store
  - Write register to memory location



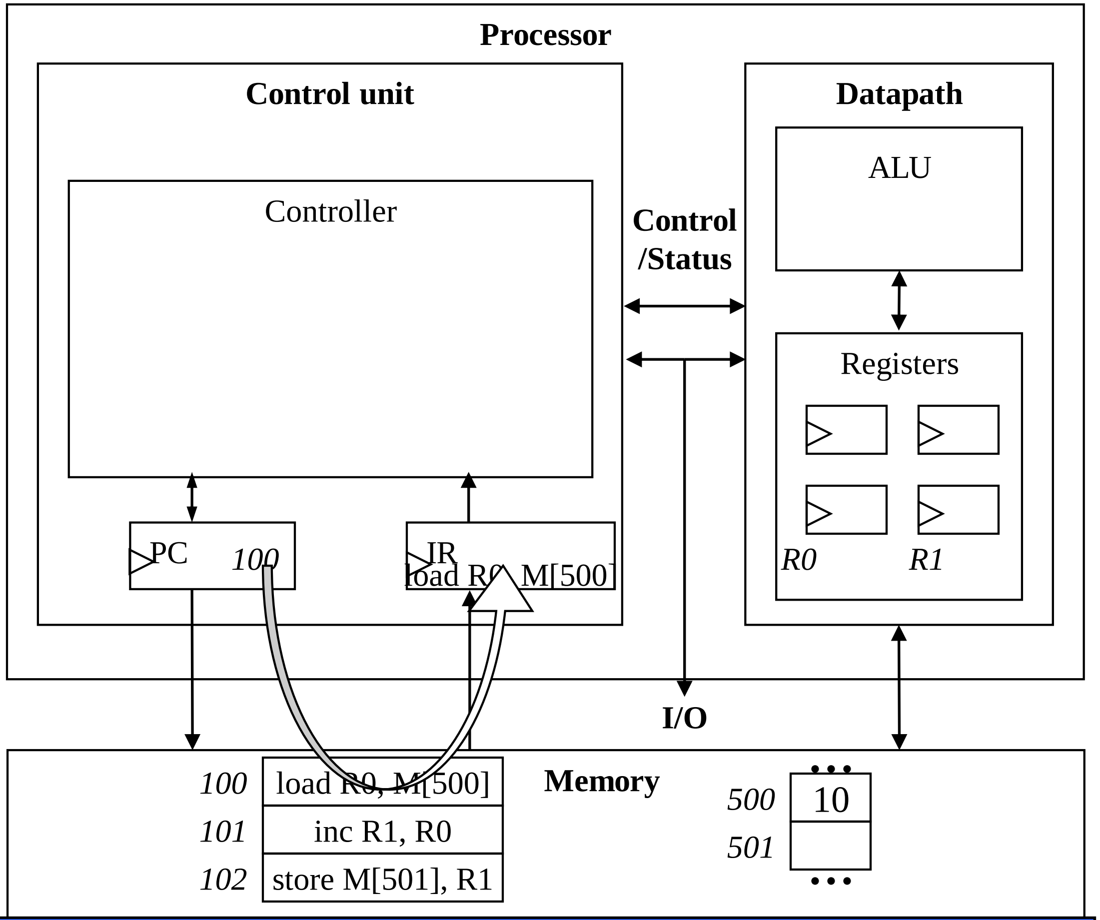
Control Unit

- Control unit: configures the datapath operations
  - Sequence of desired operations ("instructions") stored in memory – "program"
- Instruction cycle – broken into several sub-operations, each one clock cycle, e.g.:
  - Fetch: Get next instruction into IR
  - Decode: Determine what the instruction means
  - Fetch operands: Move data from memory to datapath register
  - Execute: Move data through the ALU
  - Store results: Write data from register to memory



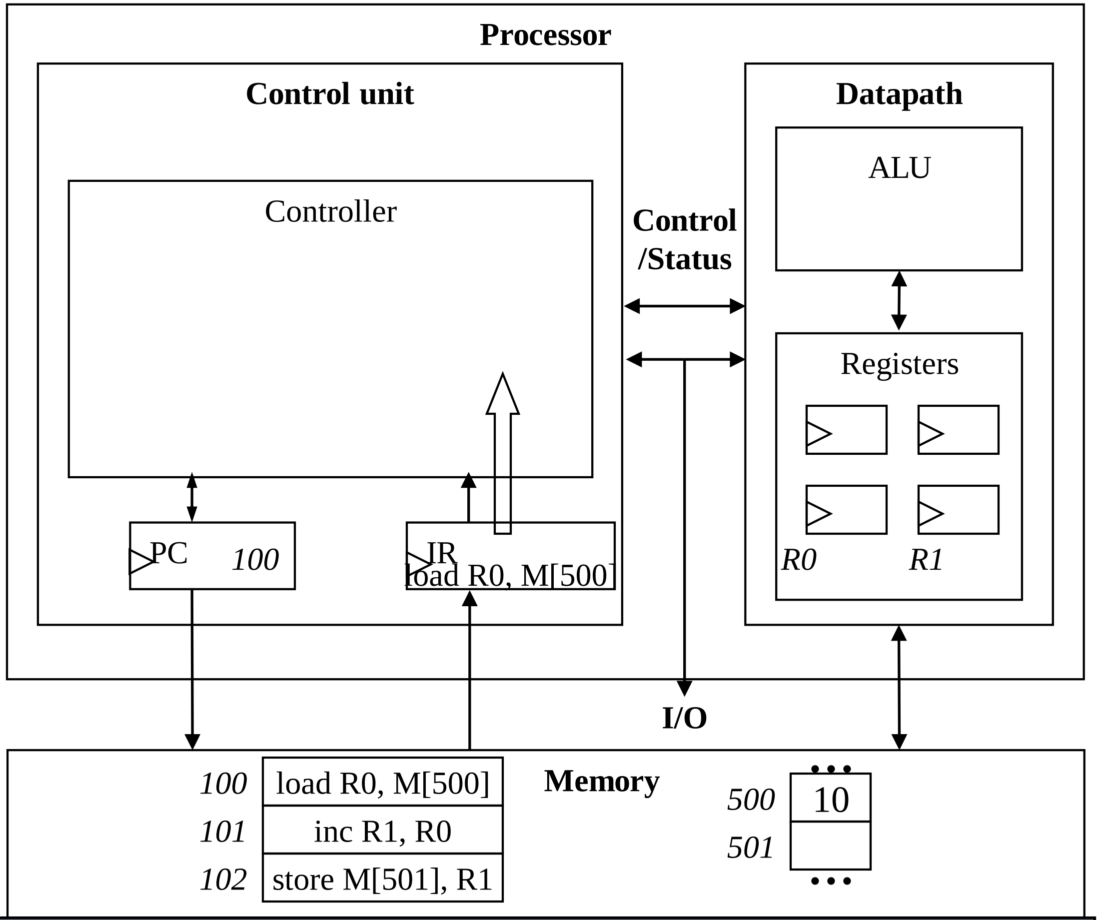
Control Unit Sub-Operations

- Fetch
  - Get next instruction into IR
  - PC: program counter, always points to next instruction
  - IR: holds the fetched instruction



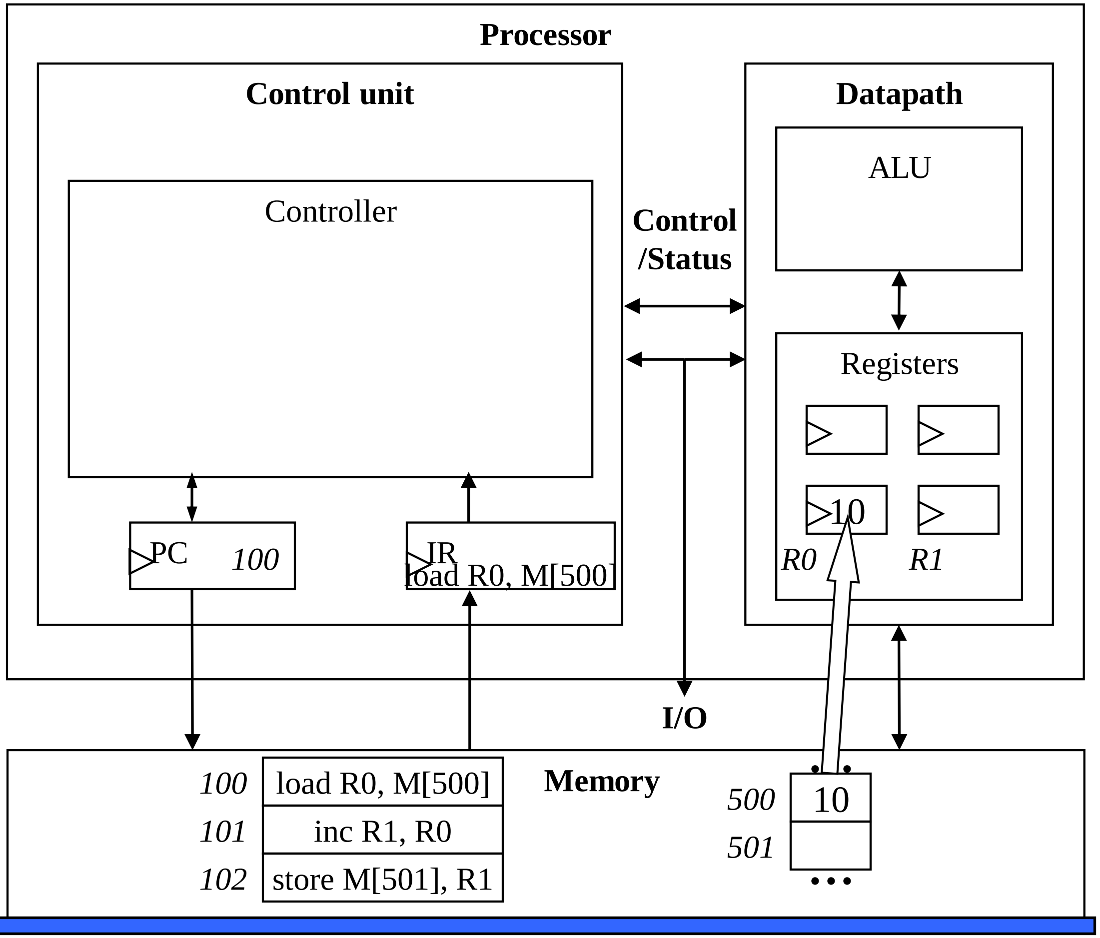
Control Unit Sub-Operations

- Decode
  - Determine what the instruction means



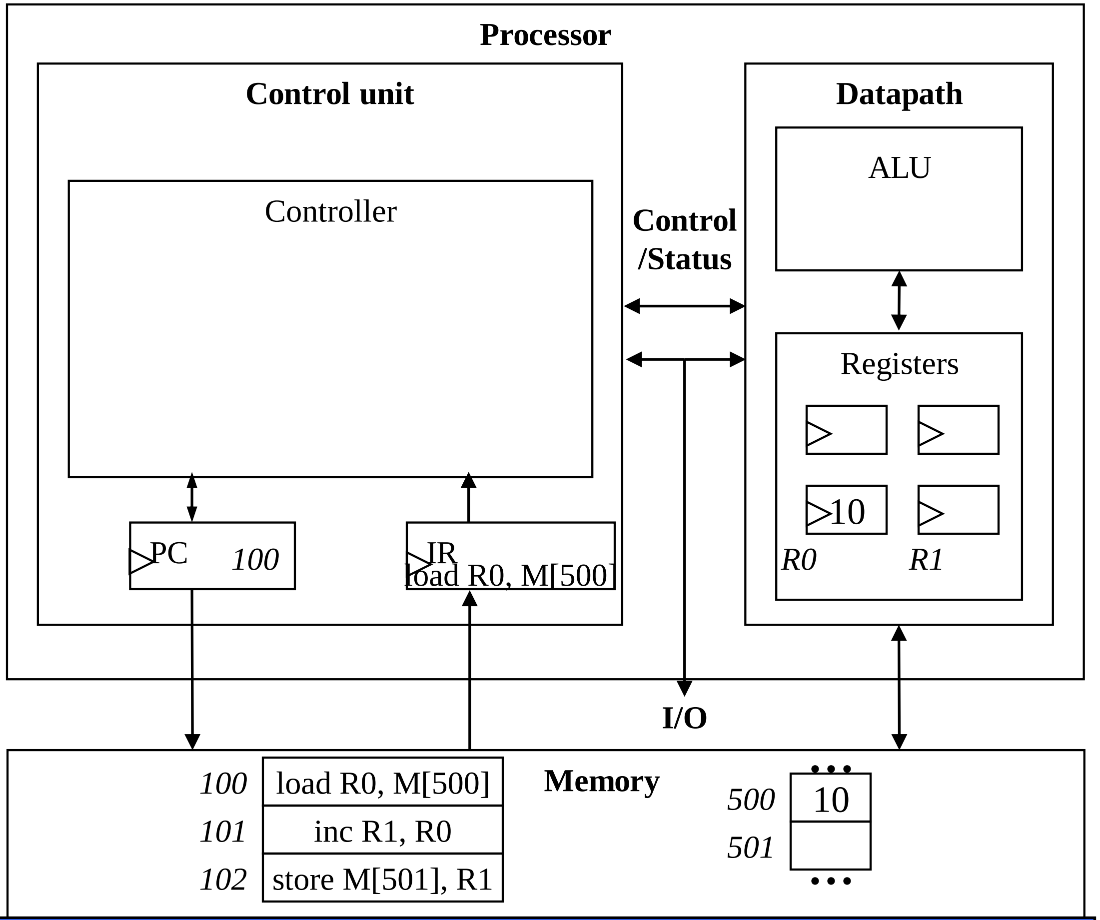
Control Unit Sub-Operations

- Fetch operands
  - Move data from memory to datapath register



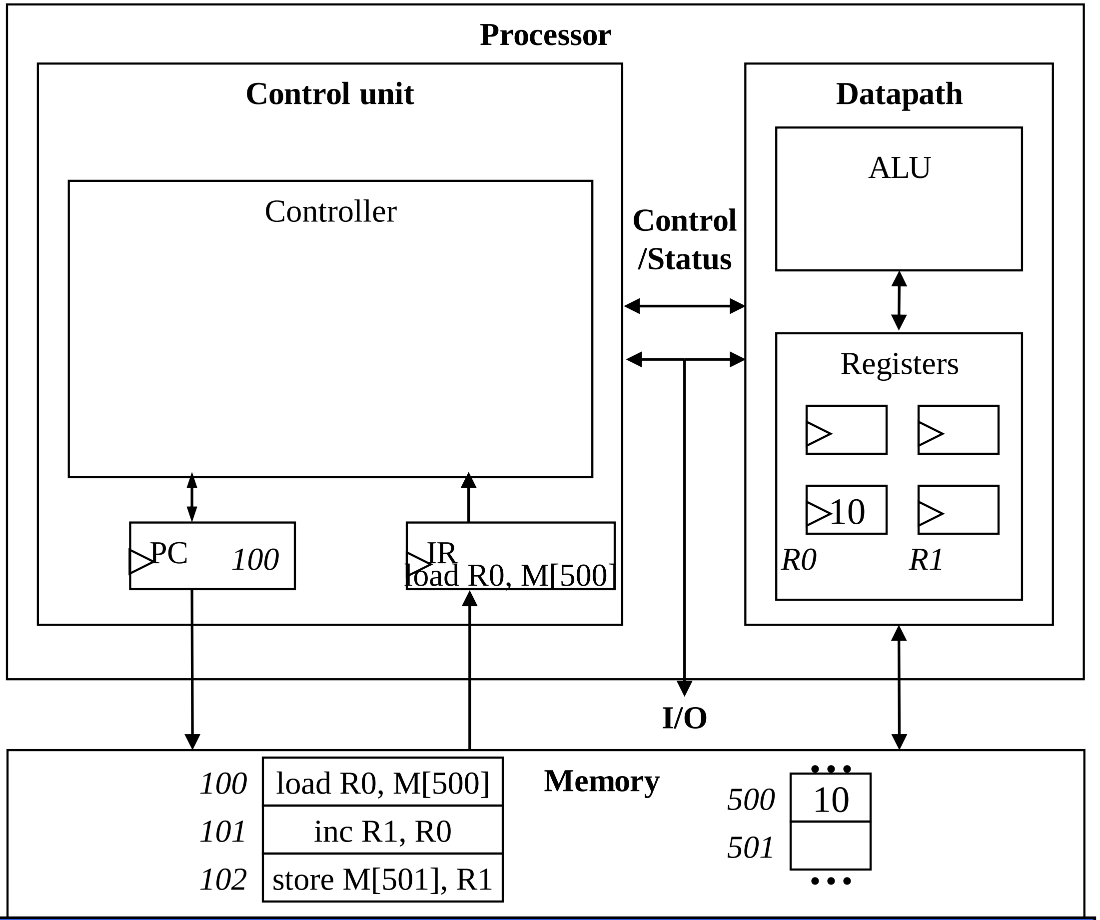
Control Unit Sub-Operations

- Execute
  - Move data through the ALU
  - This particular instruction does nothing during this sub-operation



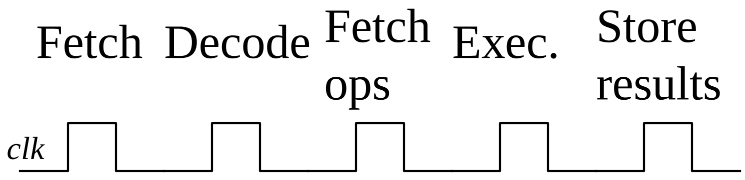
Control Unit Sub-Operations

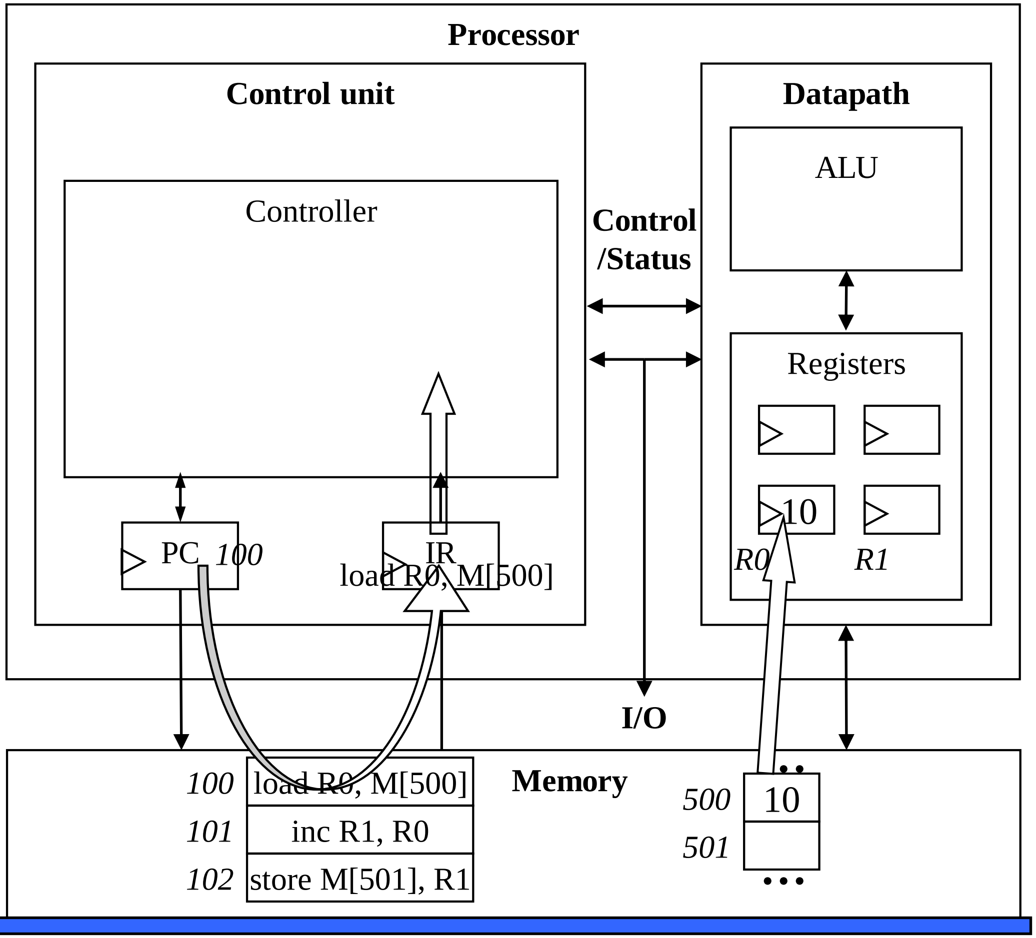
- Store results
  - Write data from register to memory
  - This particular instruction does nothing during this sub-operation



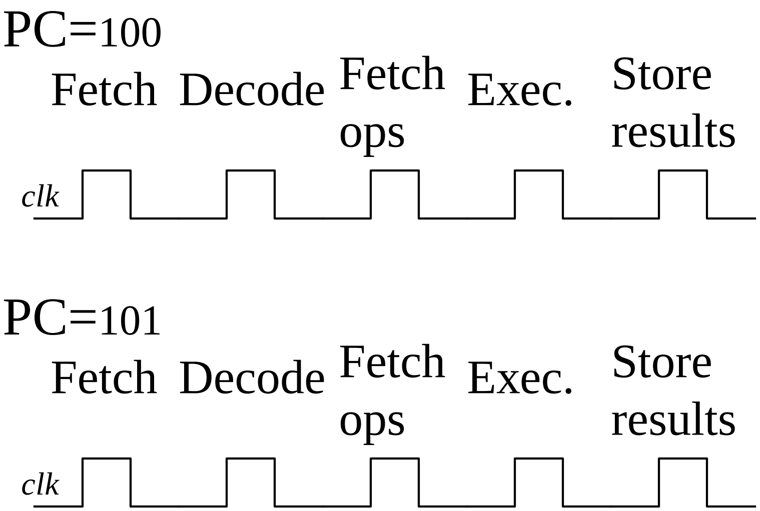
Instruction Cycles

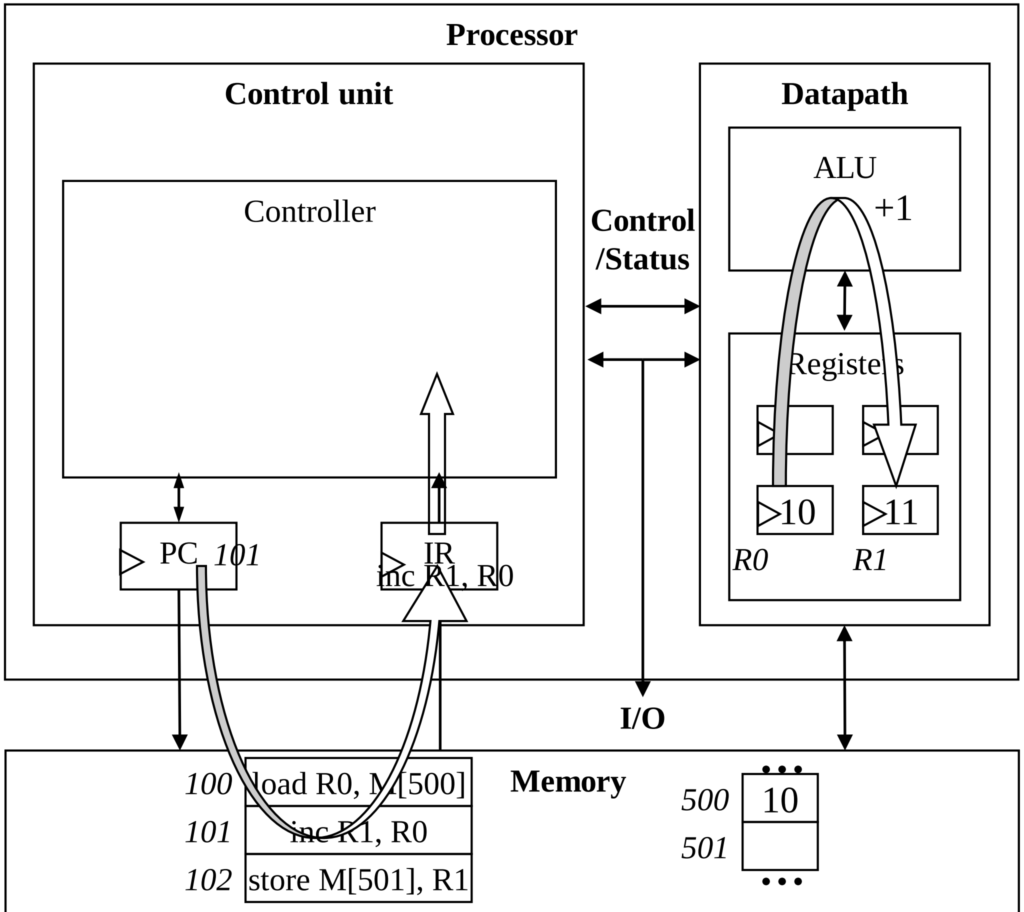
PC=100



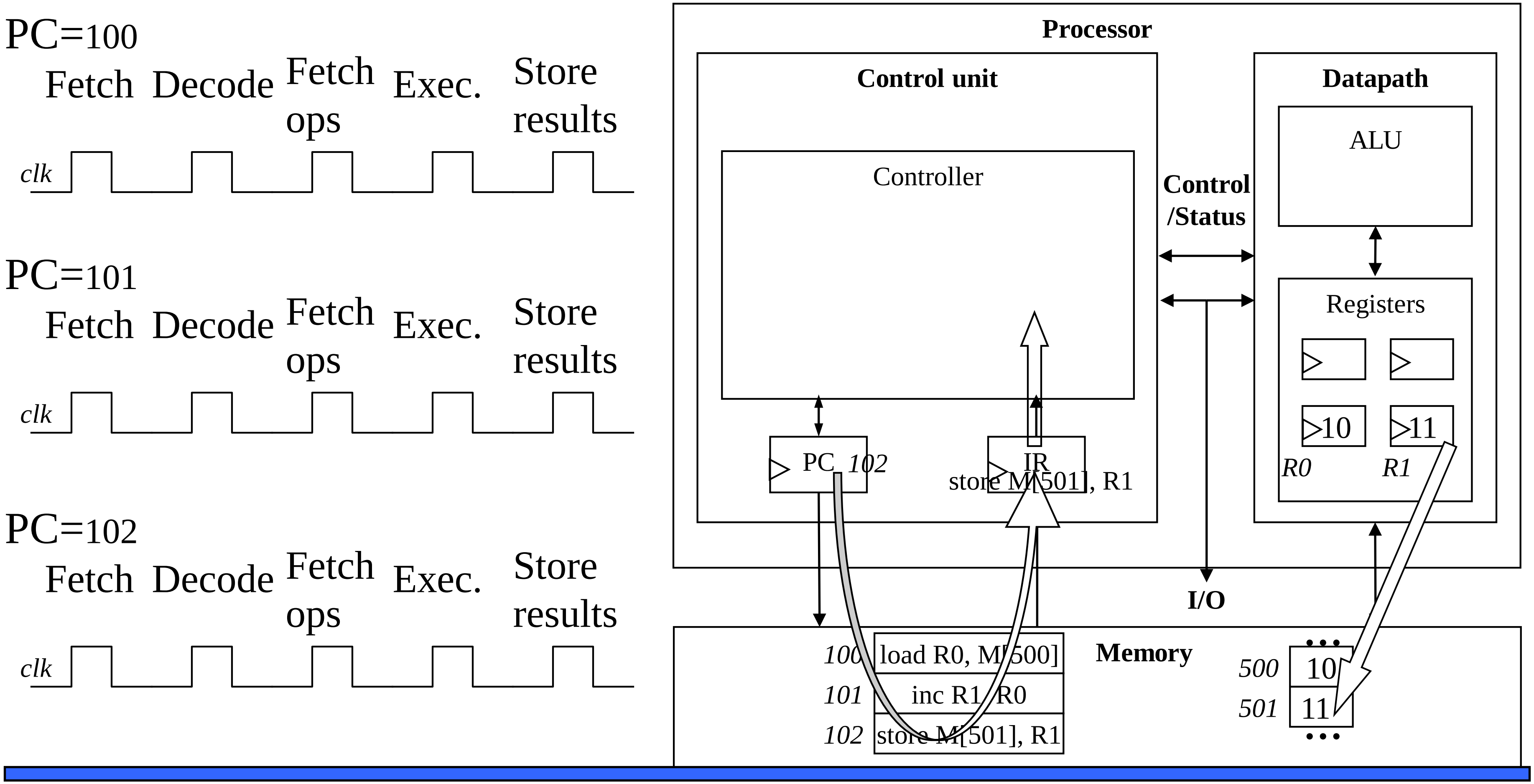


Instruction Cycles



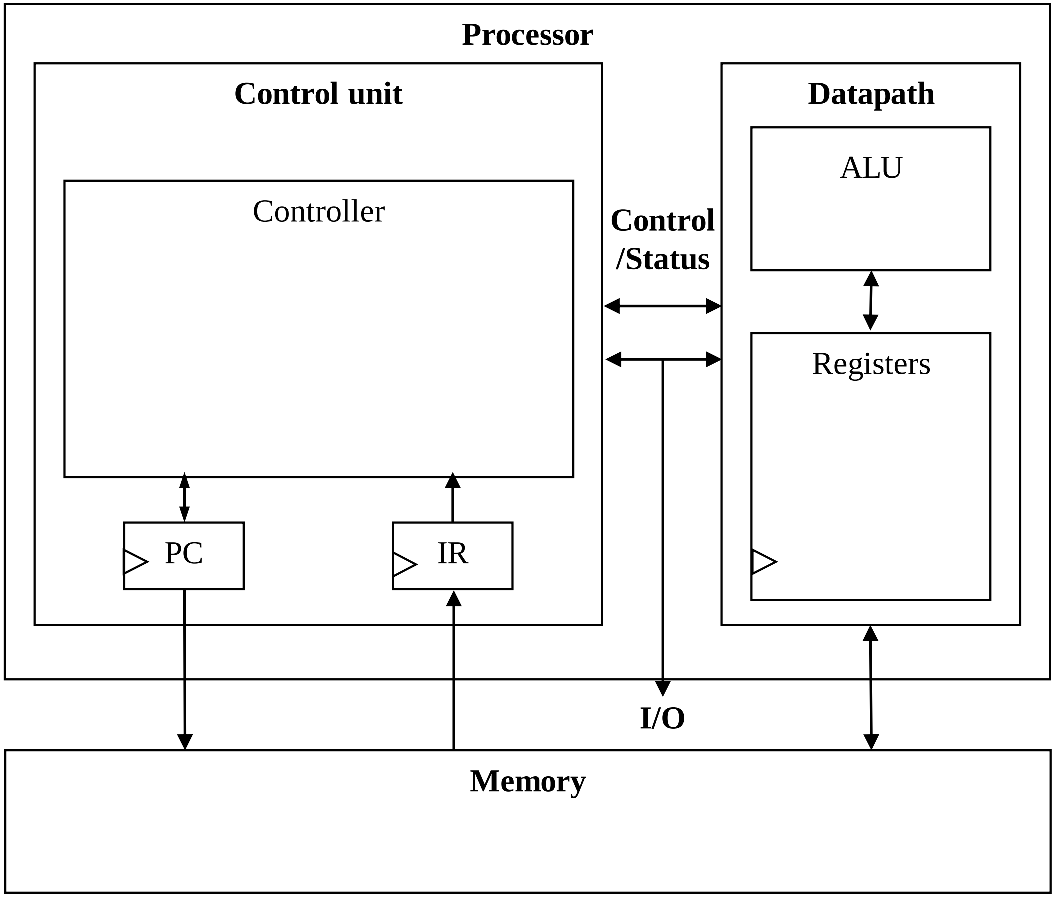


Instruction Cycles



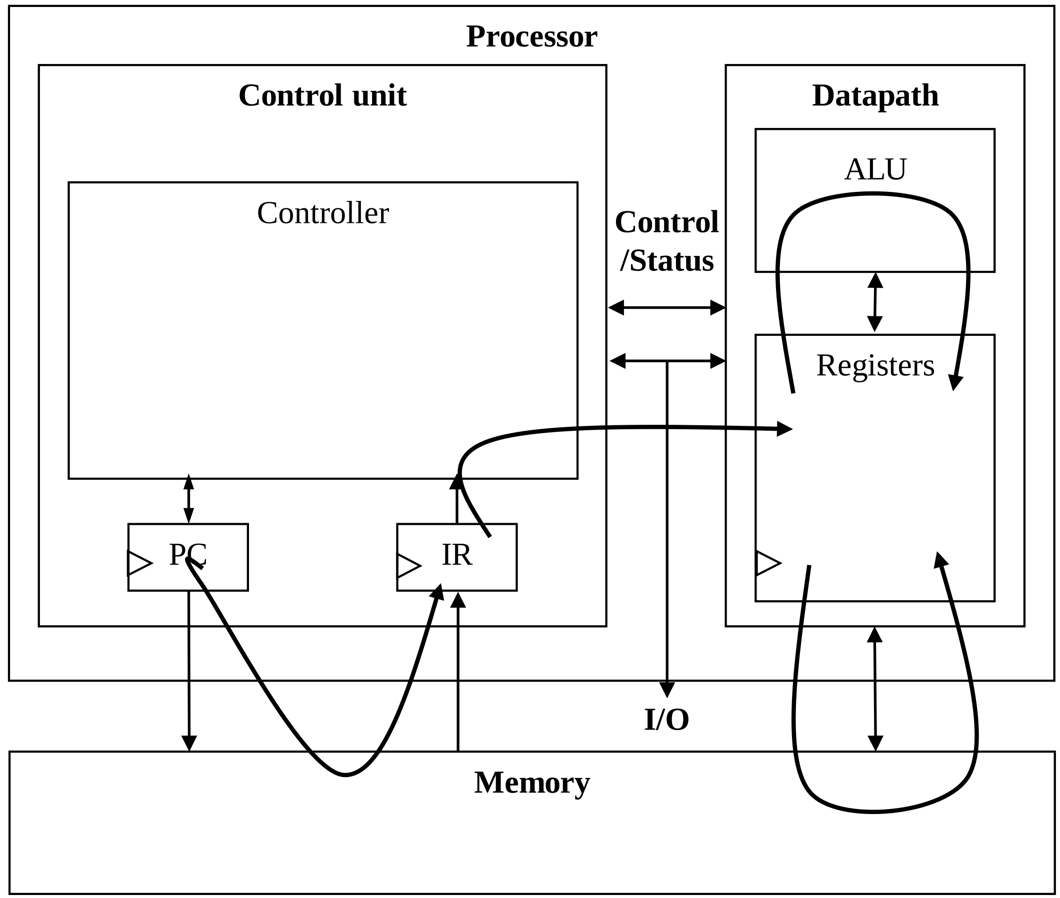
Architectural Considerations

- *N-bit* processor
  - N-bit ALU, registers, buses, memory data interface
  - Embedded: 8-bit, 16-bit, 32-bit common
  - Desktop/servers: 32-bit, even 64
- PC size determines address space



Architectural Considerations

- Clock frequency
  - Inverse of clock period
  - Clock period must be longer than required time for data to travel from one register to another in entire processor
  - Memory access is often the longest

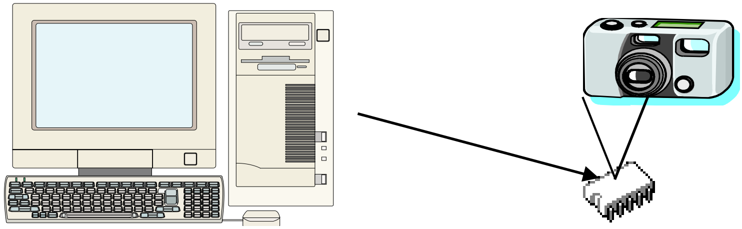


Programmer's View

- Programmer doesn't need detailed understanding of architecture
  - Instead, needs to know what instructions can be executed
- Programming Options:
  - Direct Programming (C, C++, Java, etc.)
  - Using Application Programming Interface (API)

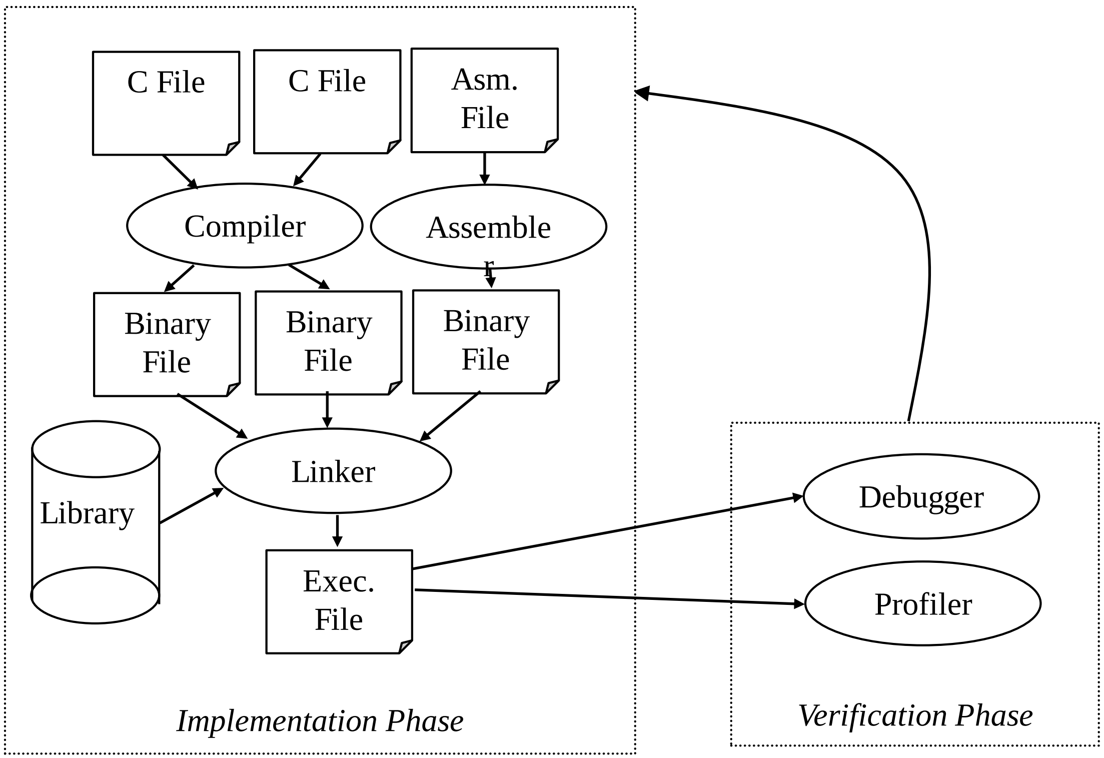
Development Environment

- Development processor
  - The processor on which we write and debug our programs
    - Usually a PC
- *Target processor*
  - The processor that the program will run on in our embedded system
    - Often different from the development processor



Development processor　　Target processor

Software Development Process

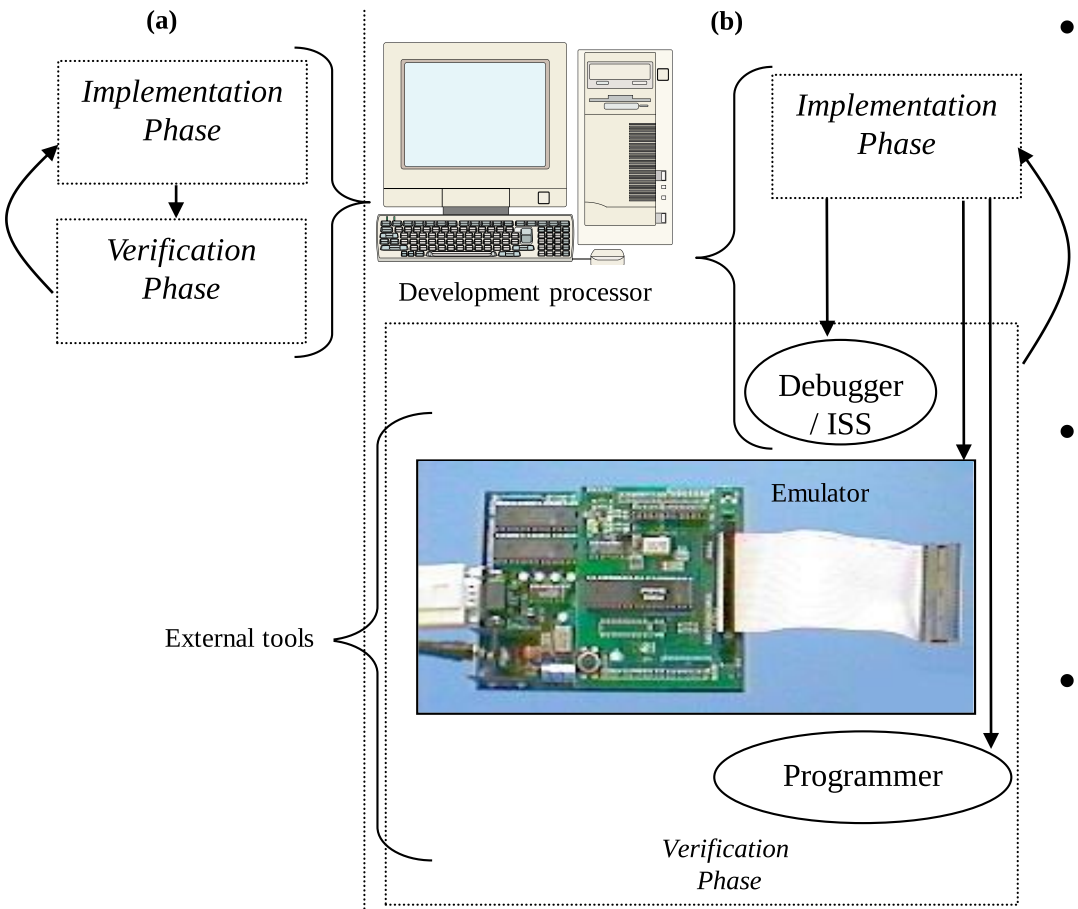


- Compilers
  - Cross compiler
    - Runs on one processor, but generates code for another
- Assemblers
- Linkers
- Debuggers
- Profilers

Running a Program

- If development processor is different than target, how can we run our compiled code? Two options:
  - Download to target processor
  - Simulate
- Simulation
  - One method: Hardware description language
    - But slow, not always available
  - Another method: *Instruction set simulator (ISS)*
    - Runs on development processor, but executes instructions of target processor

Testing and Debugging



- ISS
  - Gives us control over time – set breakpoints, look at register values, set values, step-by-step execution, ...
  - But, doesn't interact with real environment
- Download to board
  - Use device programmer
  - Runs in real environment, but not controllable
- Compromise: emulator
  - Runs in real environment, at speed or near
  - Supports some controllability from the PC

Application-Specific Instruction-Set Processors (ASIPs)

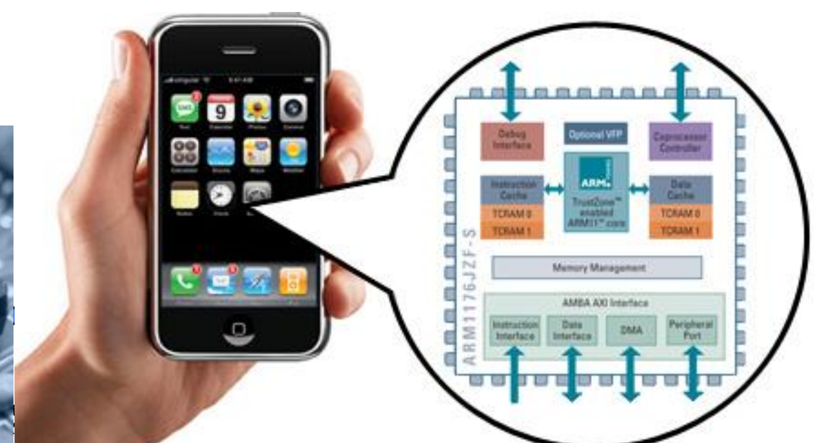
- General-purpose processors
  - Sometimes too general to be effective in demanding application
    - e.g., video processing – requires huge video buffers and operations on large arrays of data, inefficient on a GPP
  - But single-purpose processor has high NRE, not programmable
- ASIPs – targeted to a particular domain
  - Contain architectural features specific to that domain
    - e.g., embedded control, digital signal processing, video processing, network processing, telecommunications, etc.
  - Still programmable

A Common ASIP: Microcontroller

- For embedded control applications
  - Reading sensors, setting actuators
  - Mostly dealing with events (bits): data is present, but not in huge amounts
  - e.g., VCR, disk drive, digital camera (assuming SPP for image compression), washing machine, microwave oven
- Microcontroller features
  - On-chip peripherals
    - Timers, analog-digital converters, serial communication, etc.
    - Tightly integrated for programmer, typically part of register space
  - On-chip program and data memory
  - Direct programmer access to many of the chip's pins
  - Specialized instructions for bit-manipulation and other low-level operations

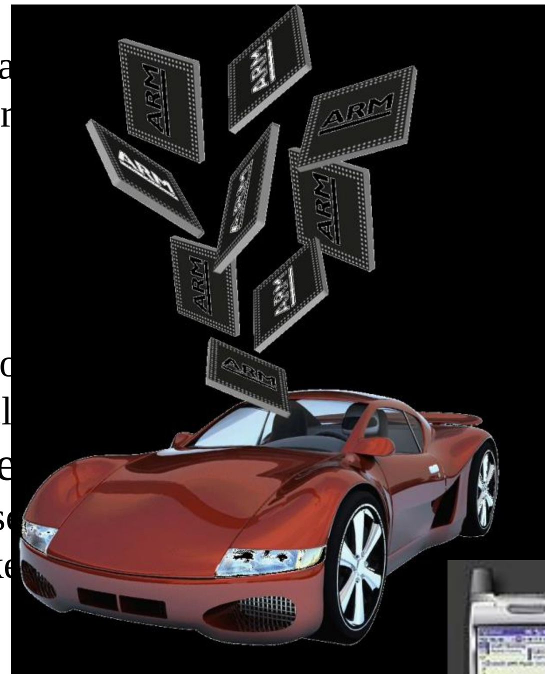
ARM-Based Devices

- As of 2013, it is the most widely used 32-bit instruction set architecture in the world
- According to ARM Holdings, in 2010 alone, producers of chips based on ARM architectures reported shipments of 6.1 billion ARM-based processors, representing 95% of smartphones, 35% of digital televisions and 10% of mobile computers.



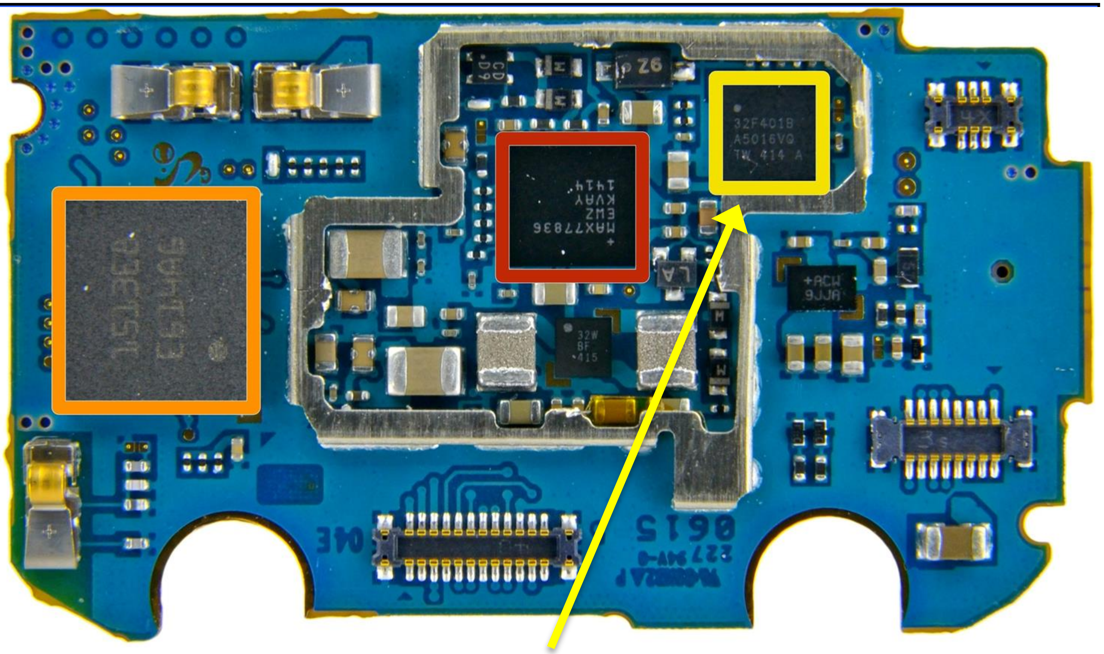
Automotive Electronics

- Vehicle infotainment
  - Radio
  - Navigation
  - Hands-free telephony
  - Voice control
  - Head-up displa
  - Back-up camer
- Powertrain
  - Engine
  - Gearbox
  - Transmission
  - Traction contro
  - Electric Vehicl
- Driver assistance
  - Dynamic cruise
  - Pre-crash brake
  - Park assist
  - Blind spot
- Chassis
  - Braking
  - Electric steering
  - Vehicle stability
  - Active suspension



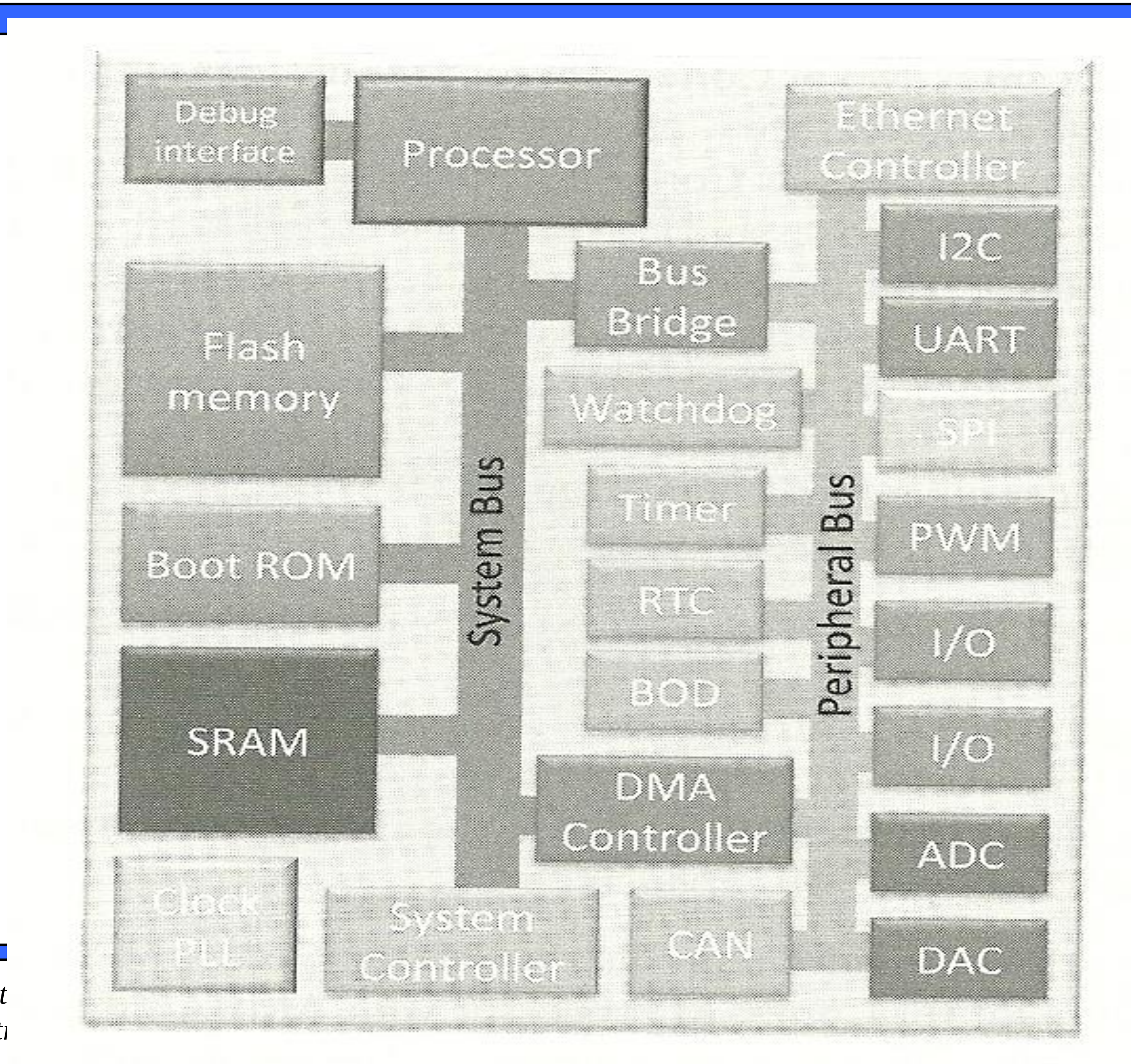
Samsung Galaxy Gear

source: *ifixit.com*



- STMicroelectronics STM32F401B **ARM-Cortex M4** MCU with 128KB Flash

Microcontroller Blocks



ARM Cortex Families

PIC Microcontroller

MICROCHIP

PIC16F84A

18-pin *Enhanced* FLASH/EEPROM 8-Bit Microcontroller

High Performance RISC CPU Features:

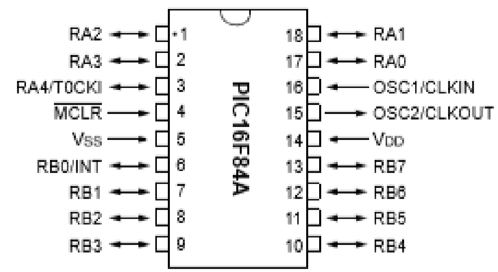
- Only 35 single word instructions to learn
- All instructions single-cycle except for program branches which are two-cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- 1024 words of program memory
- 68 bytes of Data RAM
- 64 bytes of Data EEPROM
- 14-bit wide instruction words
- 8-bit wide data bytes
- 15 Special Function Hardware registers
- Eight-level deep hardware stack
- Direct, indirect and relative addressing modes
- Four interrupt sources:
  - External RB0/INT pin
  - TMR0 timer overflow
  - PORTB<7:4> interrupt-on-change
  - Data EEPROM write complete

Peripheral Features:

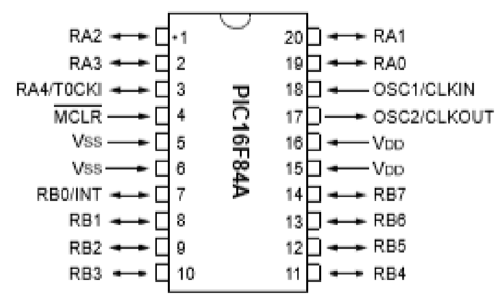
- 13 I/O pins with individual direction control
- High current sink/source for direct LED drive
  - 25 mA sink max. per pin
  - 25 mA source max. per pin
- TMR0: 8-bit timer/counter with 8-bit programmable prescaler

Pin Diagrams

PDIP, SOIC



SSOP



Selecting a Microprocessor

- Issues
  - Technical: speed, power, size, cost
  - Other: development environment, prior expertise, licensing, etc.
- Speed: how evaluate a processor's speed?
  - Clock speed – but instructions per cycle may differ
  - Instructions per second – but work per instr. may differ
  - Dhrystone: Synthetic benchmark, developed in 1984. Dhrystones/sec.
    - MIPS: 1 MIPS = 1757 Dhrystones per second (based on Digital's VAX 11/780). A.k.a. Dhrystone MIPS. Commonly used today.
      - So, 750 MIPS = 750*1757 = 1,317,750 Dhrystones per second
  - SPEC: set of more realistic benchmarks, but oriented to desktops
  - EEMBC – EDN Embedded Benchmark Consortium, www.eembc.org
    - Suites of benchmarks: automotive, consumer electronics, networking, office automation, telecommunications

General Purpose Processors

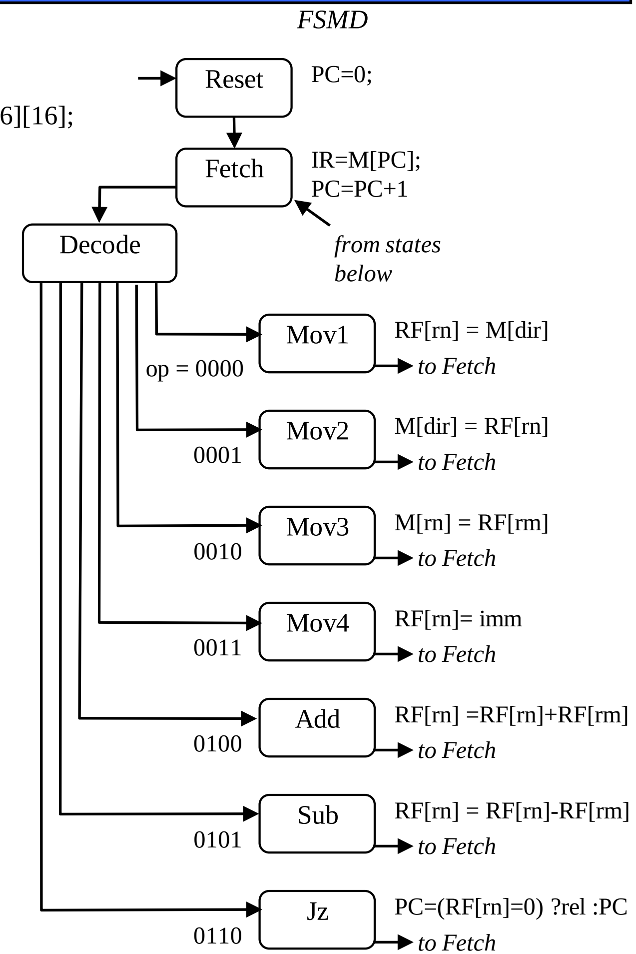
Processor	Clock speed	Periph.	Bus Width	MIPS	Power	Trans.	Price
General Purpose Processors							
Intel PIII	1GHz	2x16 K L1, 256K L2, MMX	32	~900	97W	~7M	$900
IBM PowerPC 750X	550 MHz	2x32 K L1, 256K L2	32/64	~1300	5W	~7M	$900
MIPS R5000	250 MHz	2x32 K 2 way set assoc.	32/64	NA	NA	3.6M	NA
StrongARM SA-110	233 MHz	None	32	268	1W	2.1M	NA
Microcontroller							
Intel 8051	12 MHz	4K ROM, 128 RAM, 32 I/O, Timer, UART	8	~1	~0.2W	~10K	$7
Motorola 68HC811	3 MHz	4K ROM, 192 RAM, 32 I/O, Timer, WDT, SPI	8	~.5	~0.1W	~10K	$5
Digital Signal Processors							
TI C5416	160 MHz	128K, SRAM, 3 T1 Ports, DMA, 13 ADC, 9 DAC	16/32	~600	NA	NA	$34
Lucent DSP32C	80 MHz	16K Inst., 2K Data, Serial Ports, DMA	32	40	NA	NA	$75

Sources: Intel, Motorola, MIPS, ARM, TI, and IBM Website/Datasheet; Embedded Systems Programming, Nov. 1998

Designing a General Purpose Processor

- Not something an embedded system designer normally would do
  - But instructive to see how simply we can build one top down
  - Remember that real processors aren't usually built this way
    - Much more optimized, much more bottom-up design

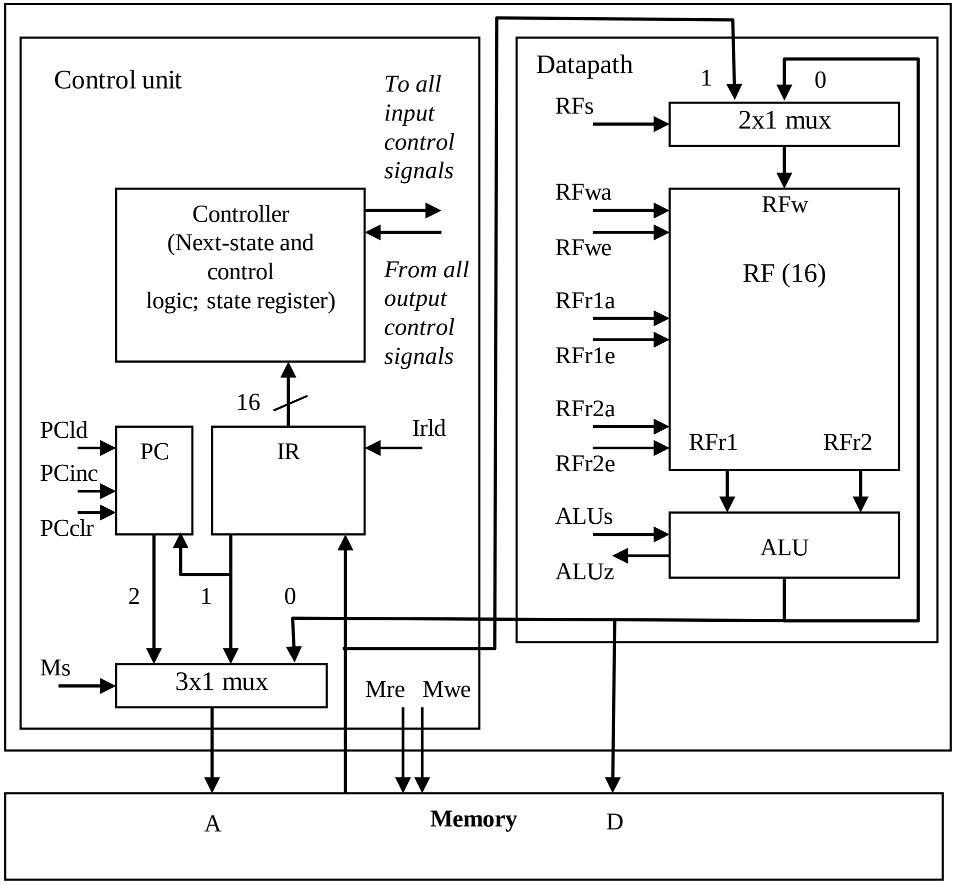
Declarations:
bit PC[16], IR[16];
bit M[64k][16], RF[16][16];



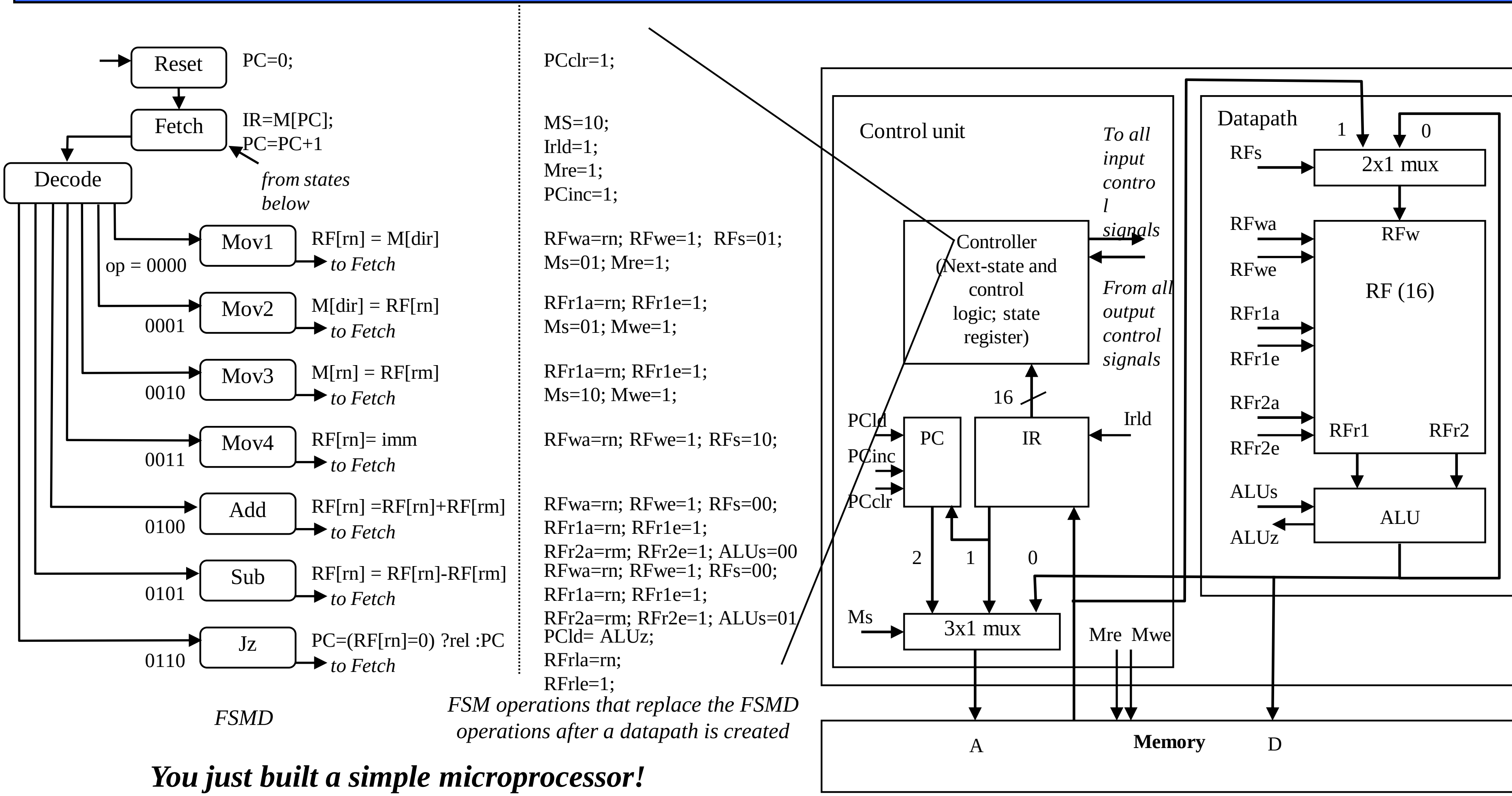
Aliases:			
op	IR[15..12]	dir	IR[7..0]
rn	IR[11..8]	imm	IR[7..0]
rm	IR[7..4]	rel	IR[7..0]

Architecture of a Simple Microprocessor

- Storage devices for each declared variable
  - register file holds each of the variables
- Functional units to carry out the FSMD operations
  - One ALU carries out every required operation
- Connections added among the components' ports corresponding to the operations required by the FSM
- Unique identifiers created for every control signal



A Simple Microprocessor



You just built a simple microprocessor!

Chapter Summary

- General-purpose processors
  - Good performance, low NRE, flexible
- Controller, datapath, and memory
- Many tools available
  - Including instruction-set simulators, and in-circuit emulators
- Options for programming general purpose processor
  - Direct programming, it is more efficient
  - Using API, it is more convenient
- ASIPs
  - Microcontrollers, DSPs, network processors, more customized ASIPs
- Choosing among general-purpose processors is an important step.