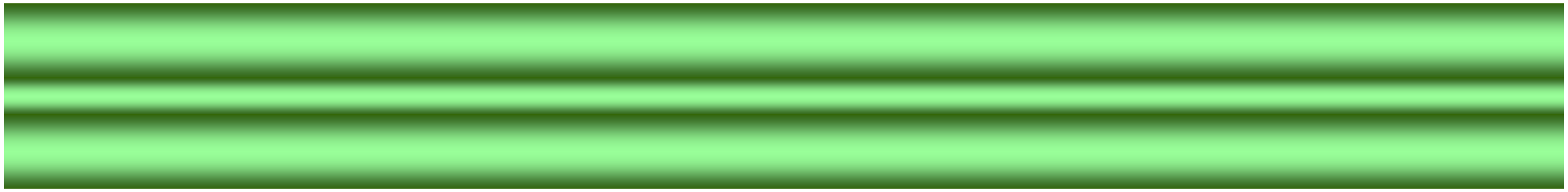
EE414 Embedded Systems

Ch 3. General-Purpose Processors: Software

Part 1/4



Byung Kook Kim
School of Electrical Engineering
Korea Advanced Institute of Science and Technology

Overview

Part 1

- 3.1 Introduction
- 3.2 Basic Architecture
- 3.3 Operation

Part 2

- 3.4 Programmer's View
- 3.5 Development Environment
- 3.6 Application-Specific Instruction-Set Processors (ASIPs)
- 3.7 Selecting a Microprocessor
- 3.8 General-Purpose Processor Design

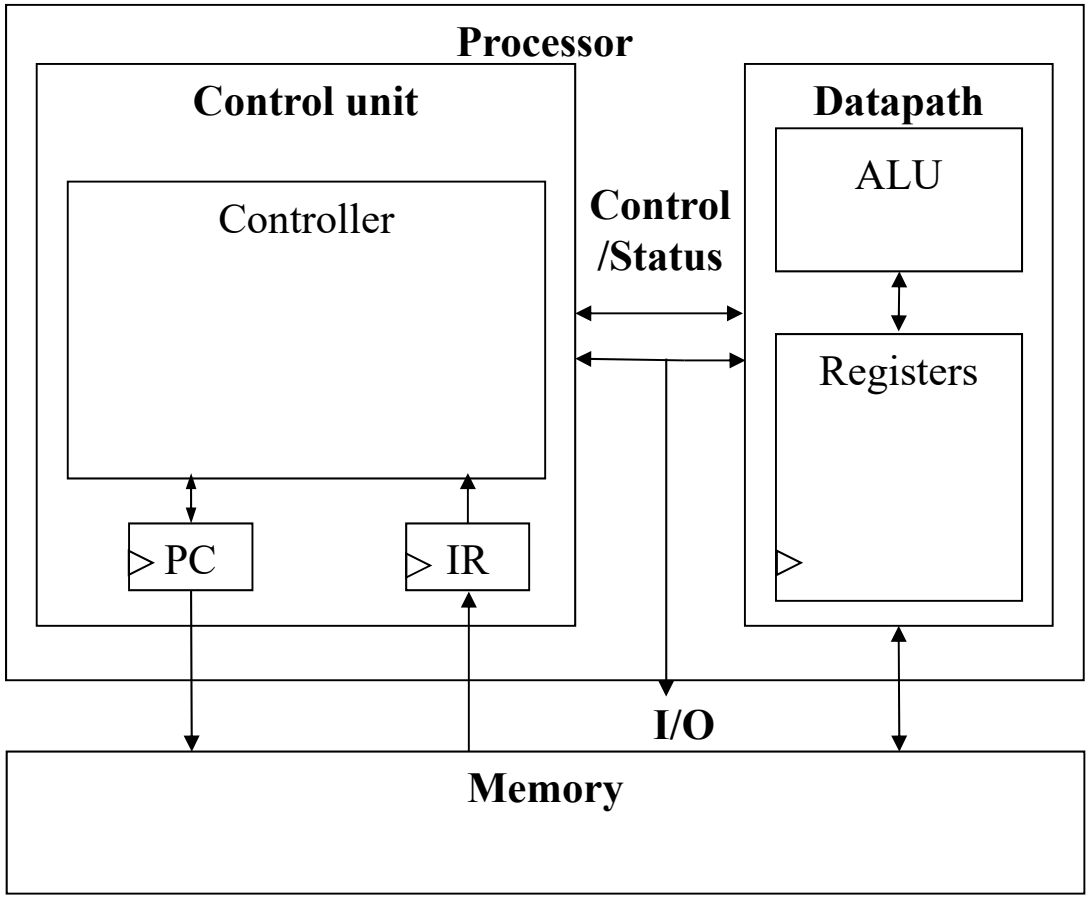
3.1 Introduction

- **General-Purpose Processor**
  - Processor designed for a variety of computation tasks
- Adv. of GPP
  - 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.
- Disadv.
  - Cost, power, size
- COTS (Commercially Off-The-Shelf) "microprocessor"
  - Processor implemented on one chip.

3.2 Basic Architecture

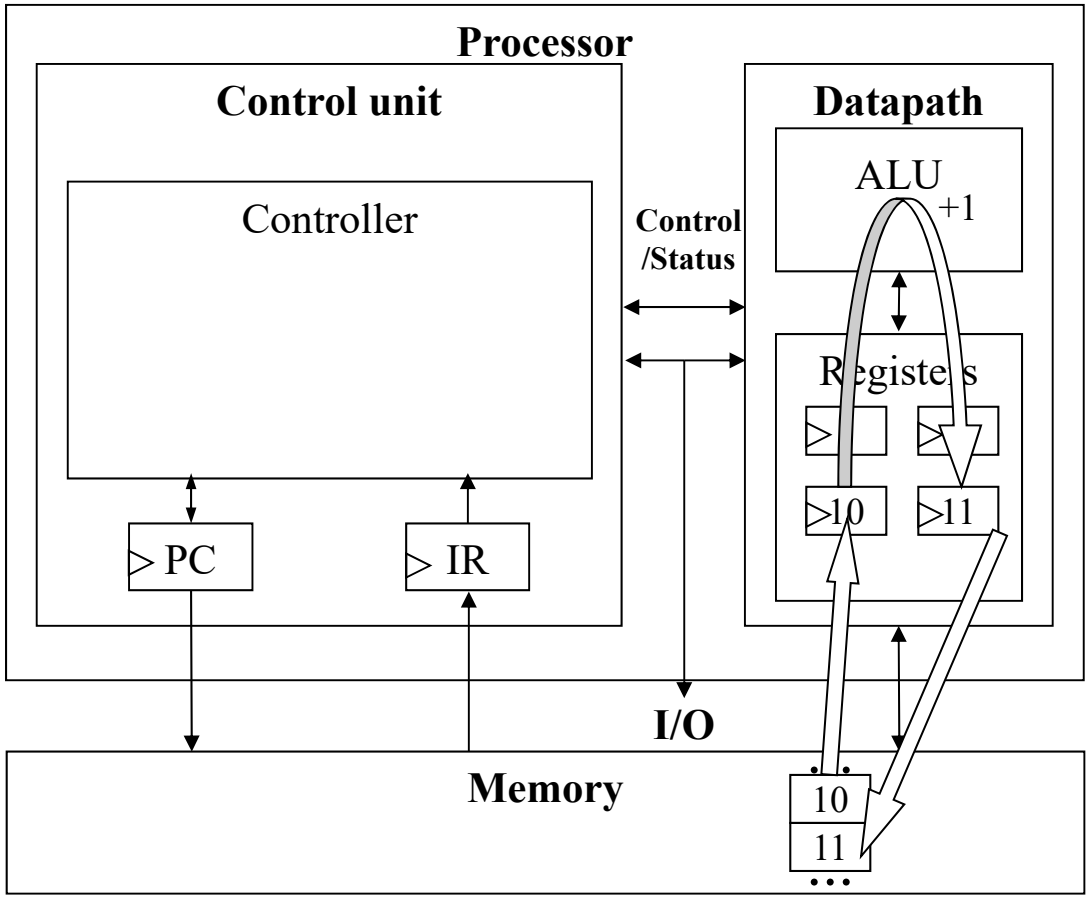
- **General-purpose processor (CPU)**: **Control unit** and **datapath,** tightly linked with a **memory**.

- Key differences to SPP
  - Datapath is general.
  - Control unit doesn't store the algorithm – the algorithm is "programmed" into the memory.
  - PC & IR



A. Datapath - Operations

- **ALU (Arithmetic and Logic Unit)**
- Load
  - Read memory location into register ALU operation
- Arithmetic & logic operation
  - Input certain registers through ALU
  - Do operation
  - Store back in register
- Store
  - Write register to memory location.
- Ex: Load, Inc, Store
- Size N: N-bit-wide



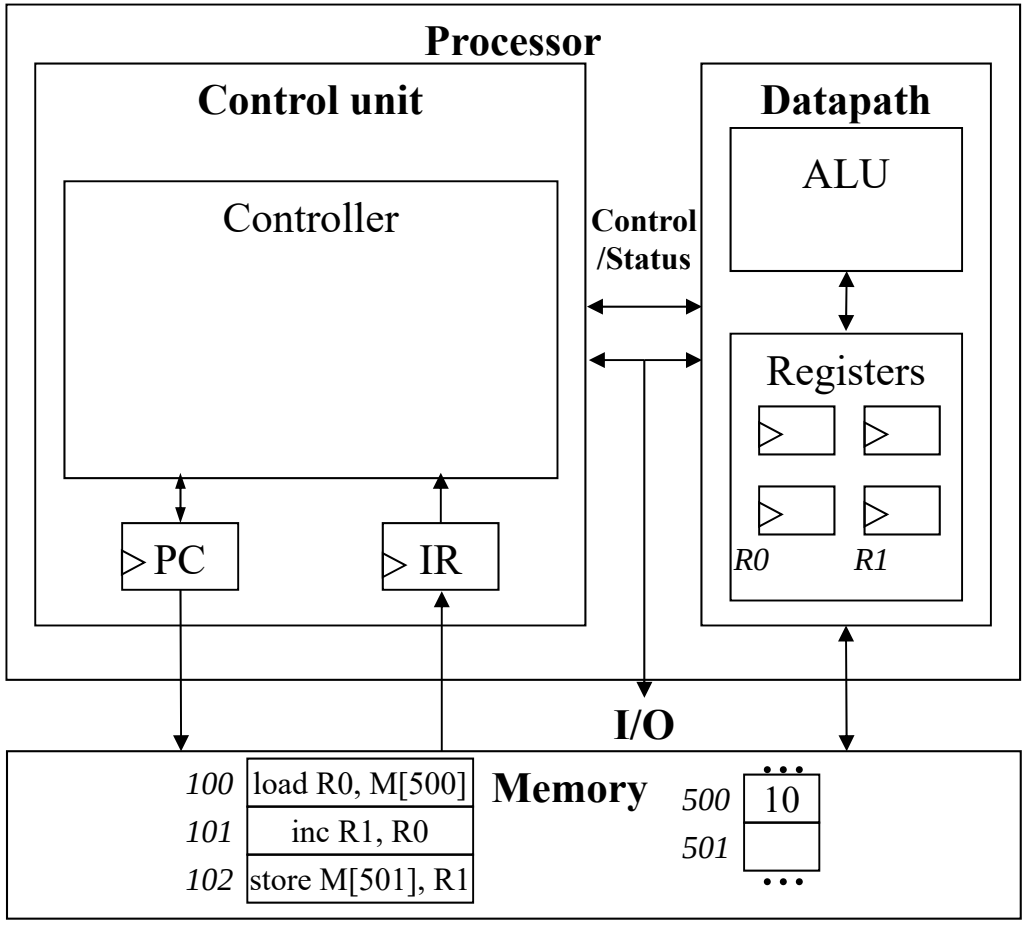
Datapath

Size of datapath operation (N-bit computer)

- 8 bits
  - Character 0 to 255 (0 to FFh/0xFF). 256 English characters.
- 16 bits
  - Integer -32768 to 32767 (-2^{15} to $2^{15}-1$).
  - Hangul characters.
- 32 bits
  - Integer -2G to 2G ($-2\text{x}10^{9}$ to $2\text{x}10^{9}$, -2^{31} to $2^{31}-1$).
  - Floating point number (Smx2S^{se}).
- 64 bits
  - Integer -8E to 8E (exa. $-8\text{x}10^{18}$ to $8\text{x}10^{18}$).
  - Double precision floating point number (SMx2^{se}).

B. Control Unit

- **Control unit**: configures the datapath operations
  - Sequence of desired operations ("instructions") stored in memory – "program"
- **Program**
  - Sequence of instructions including loop and branch.
- **Instruction**
  - Set of meaningful bits to do some operation.
  - **Machine language**
    - Binary code
      - Ex: 0011 | 00 | 1000 0001
  - **Assembly language**
    - Symbolic expression
      - Ex: LD R0, M(513)
    - 1:1 to machine language



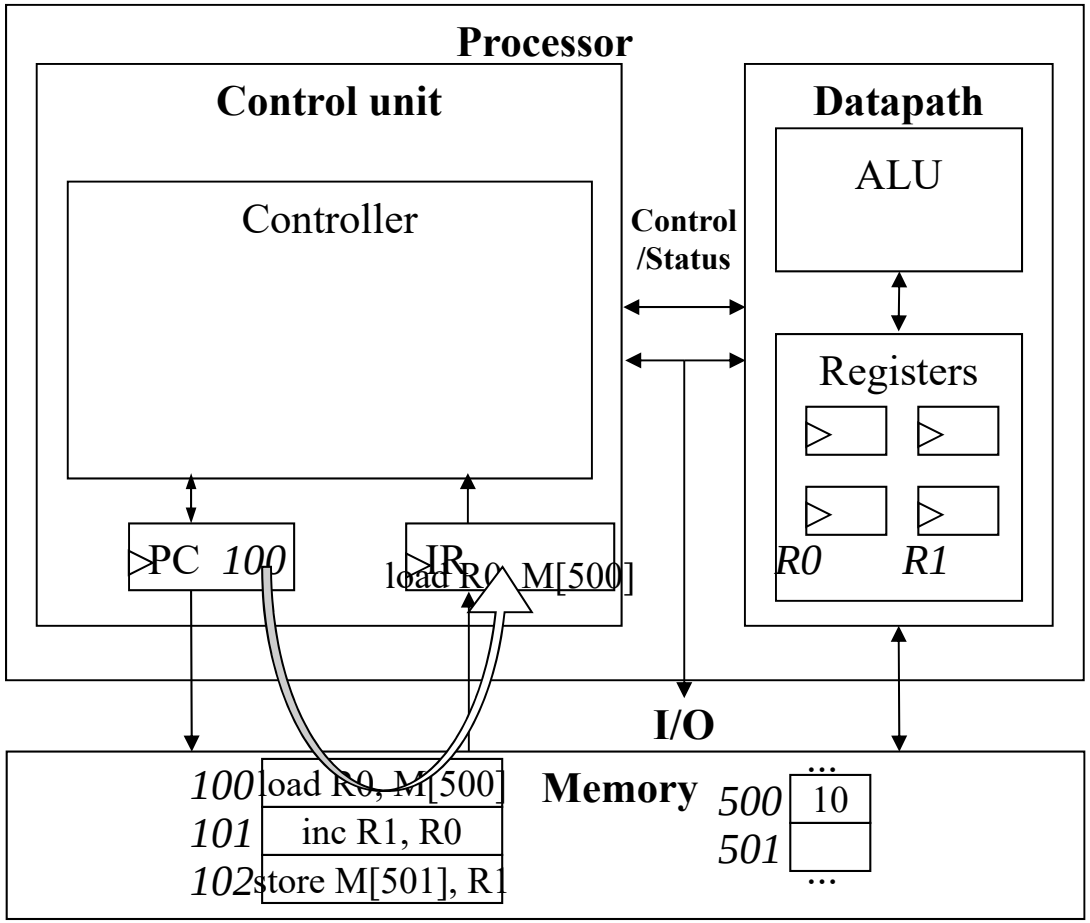
Control Unit

Instruction cycle

- Broken into several sub-operations, each one clock cycle, e.g.:
  - **1. Fetch**: Get next instruction into IR
  - **2. Decode**: Determine what the instruction means
  - **3. Fetch operands**: Move data from memory to datapath register
  - **4. Execute**: Move data through the ALU to reg
  - **5. Store results**: Write data from register to memory

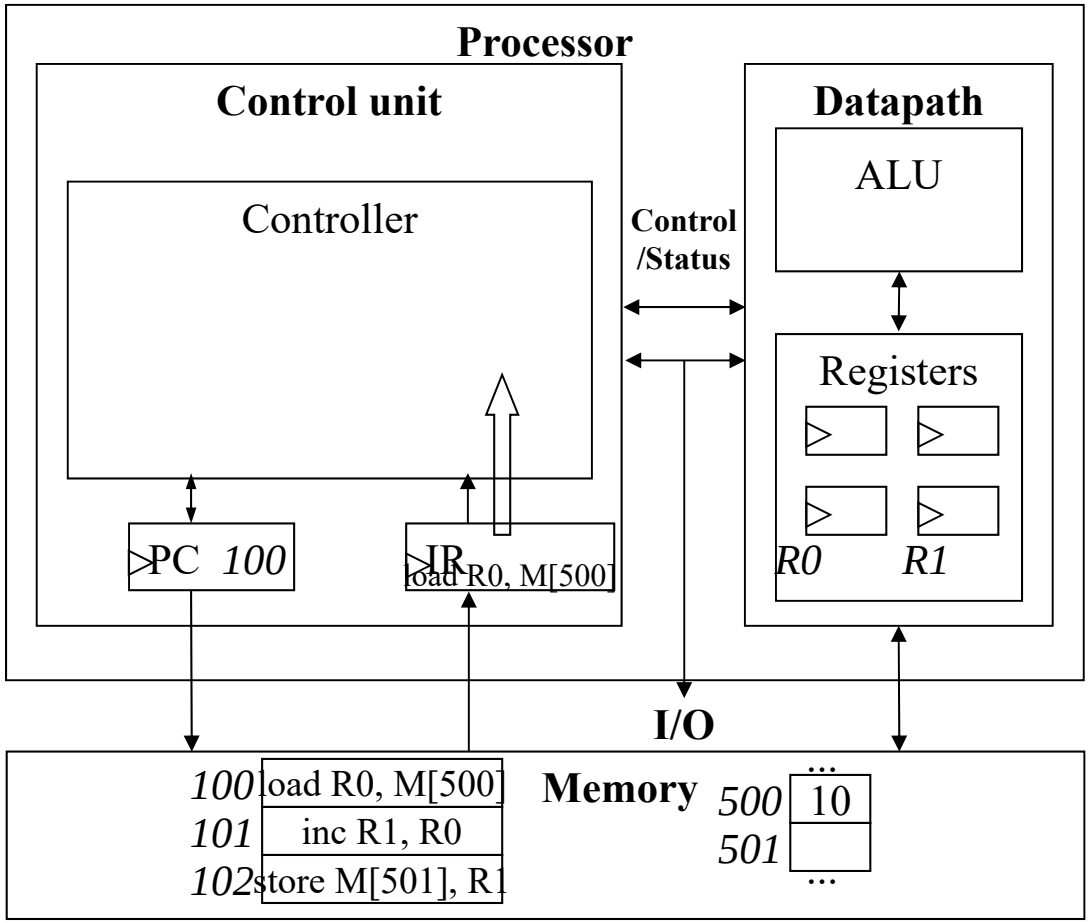
Control Unit Sub-Operation 1

- **1. Fetch**
  - Get next instruction into IR
  - **PC**: Program Counter
    - Always points to next instruction
    - Register with increment
  - **IR**: Instruction Register
    - Holds the fetched instruction.



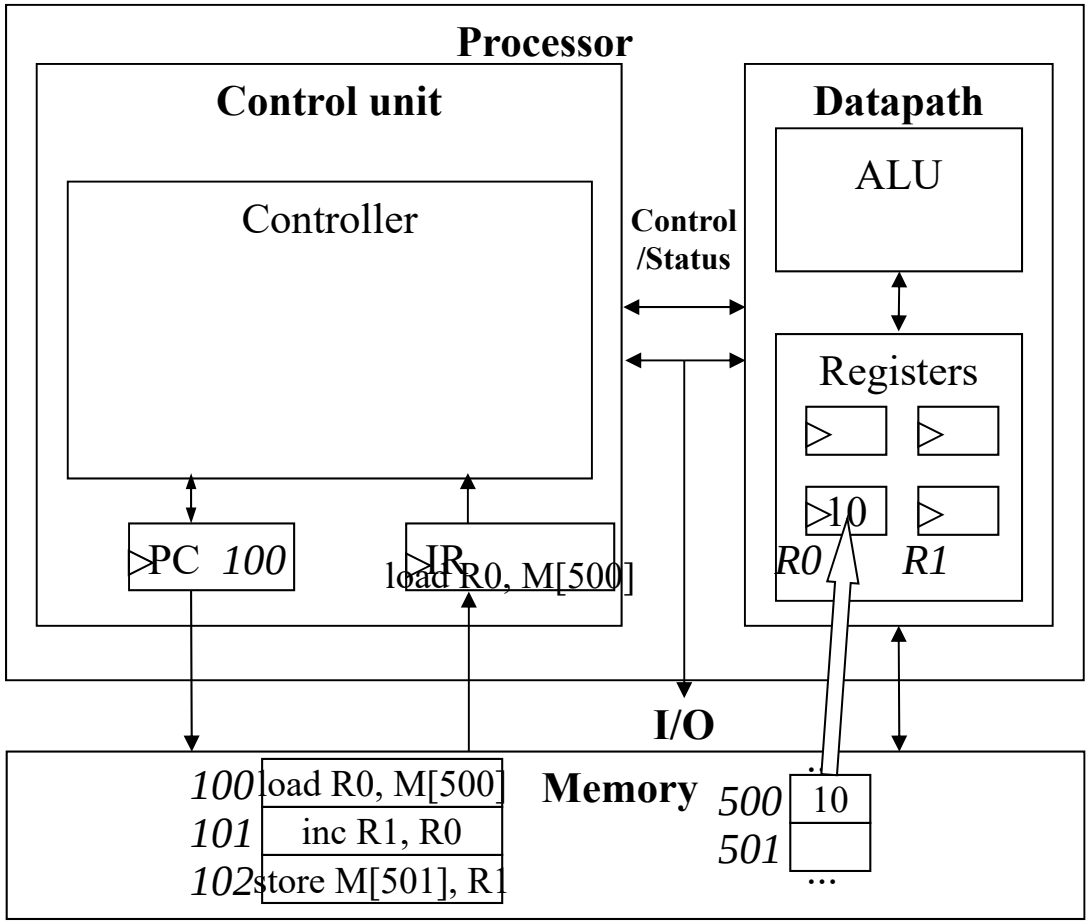
Control Unit Sub-Operation 2

- **2. Decode**
  - Determine what the instruction means
  - Using combinational logic



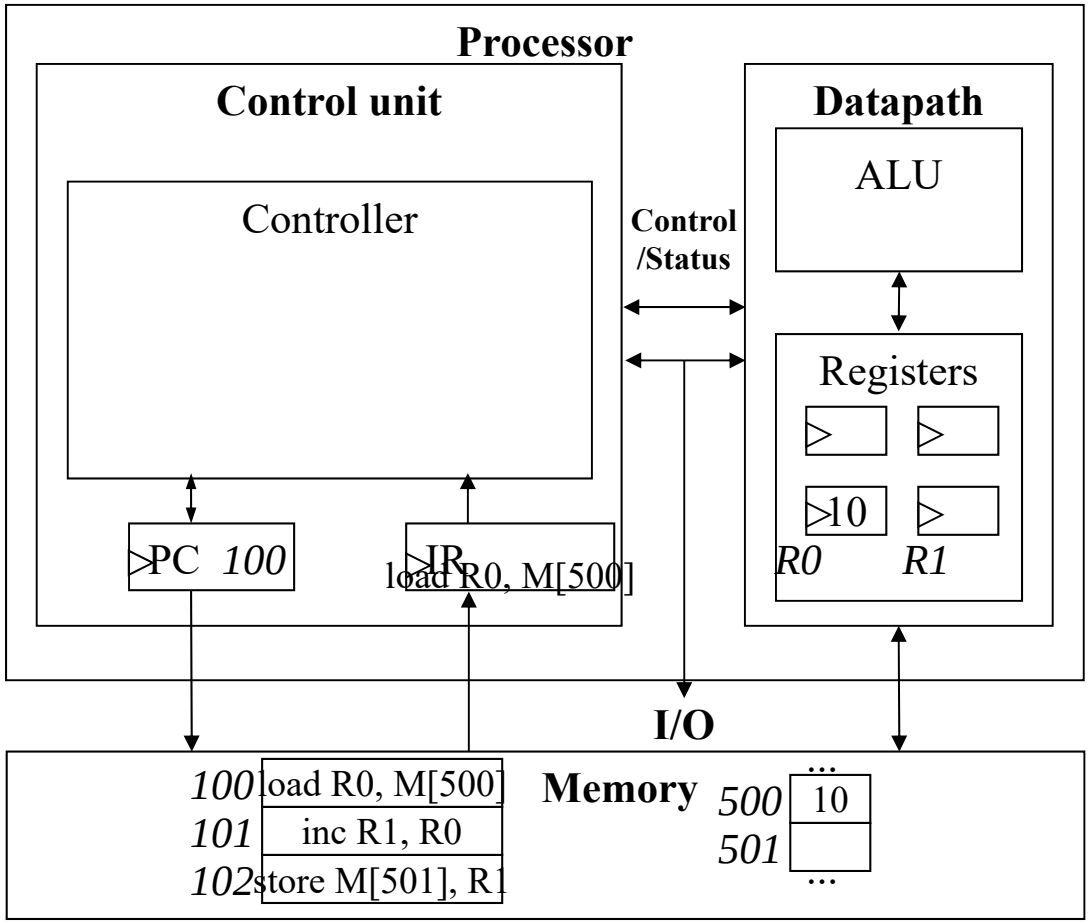
Control Unit Sub-Operation 3

- **3. Fetch operands**
  - Move data from memory to datapath register
  - Read data memory



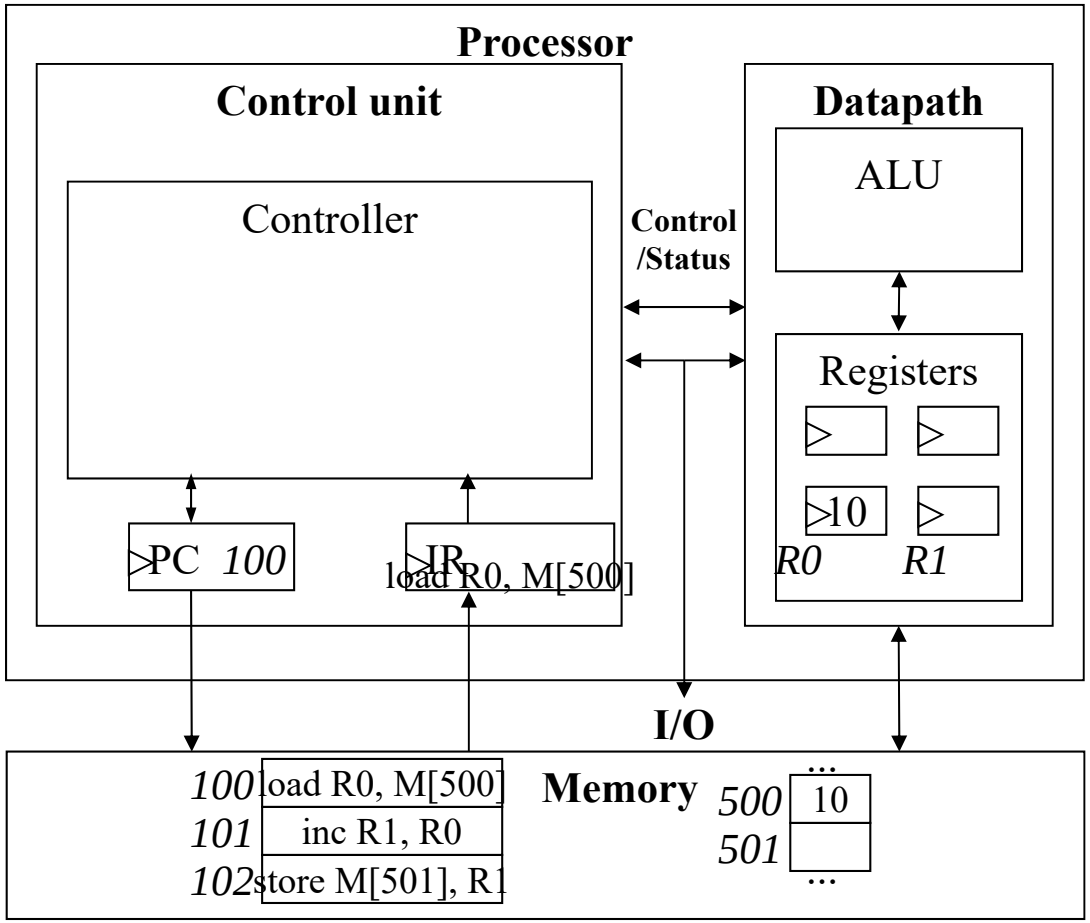
Control Unit Sub-Operation 4

- **4. Execute**
  - Move data through the ALU
  - This particular instruction LOAD does nothing during this sub-operation
  - INC
    - (R0 +1) ->R1

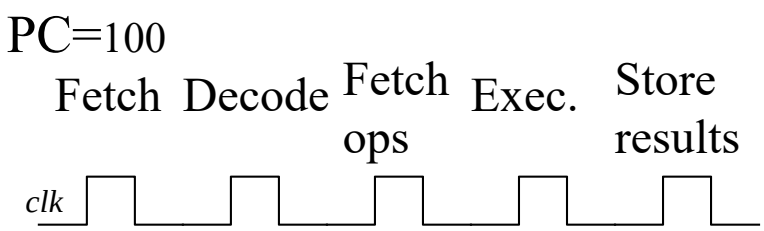


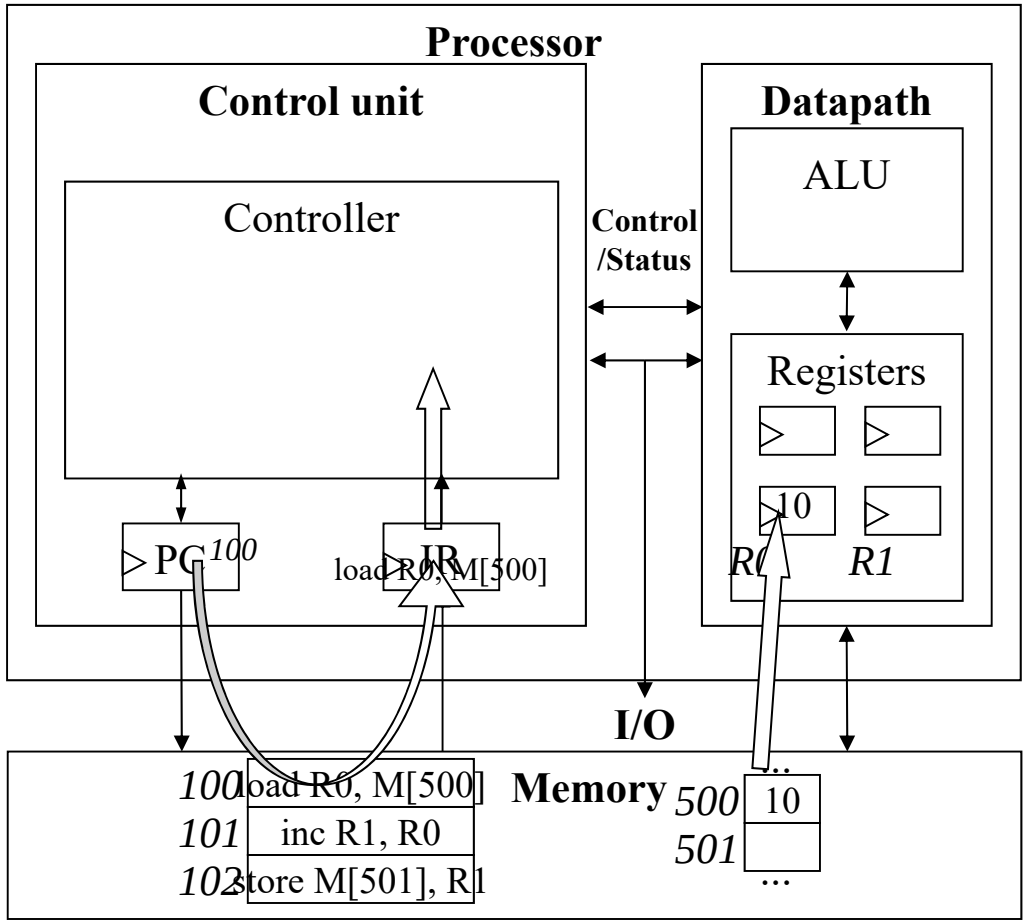
Control Unit Sub-Operation 5

- **5. Store results**
  - Write data from register to memory
  - This particular instruction LOAD does nothing during this sub-operation
  - STORE
    - R1 ->m[501]

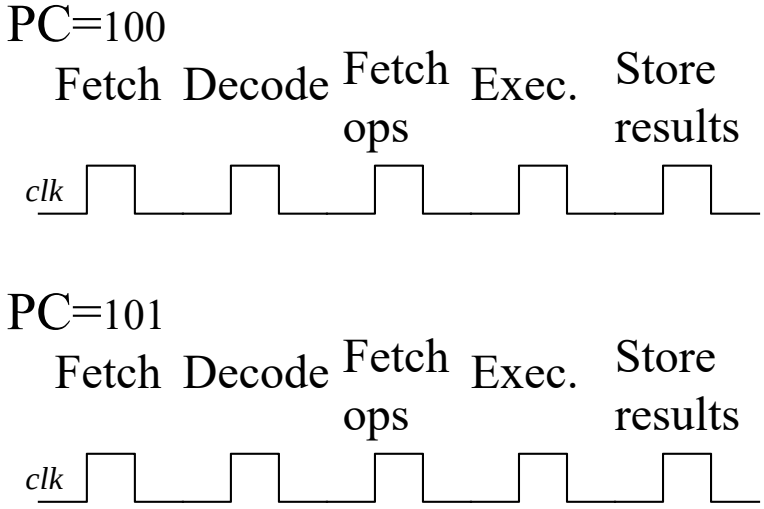


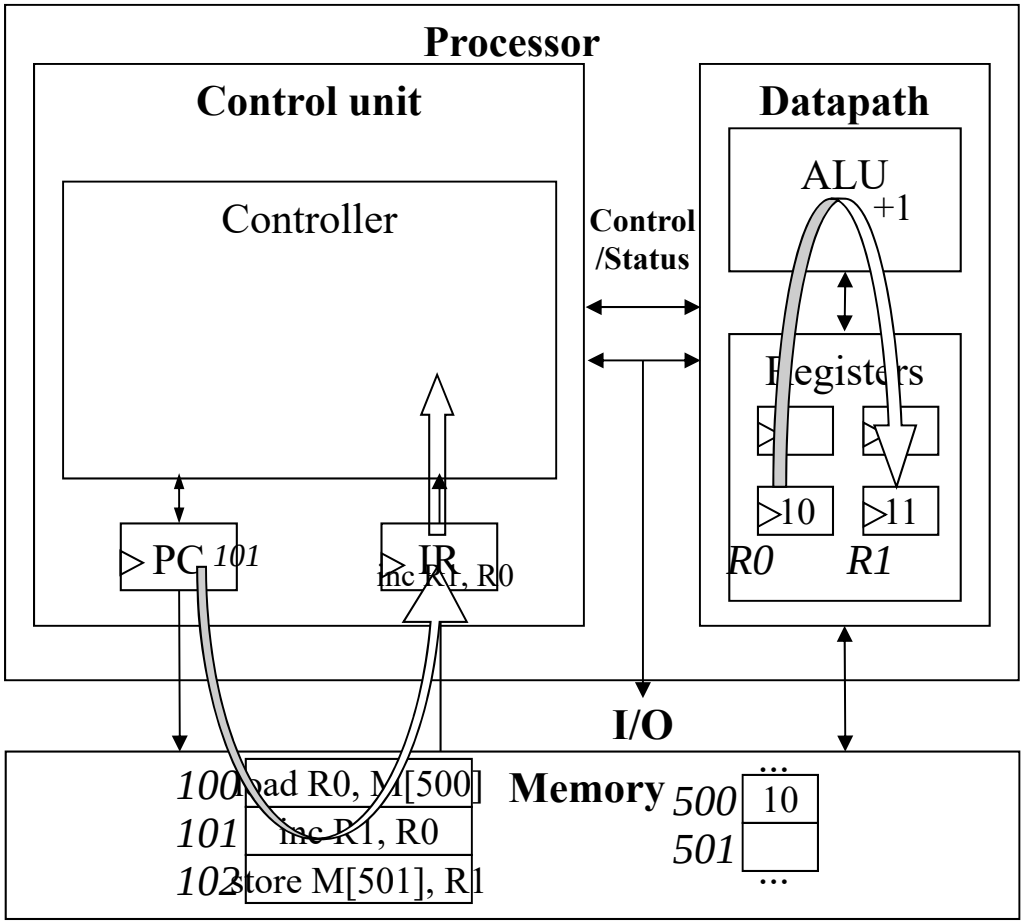
C. Instruction Cycles



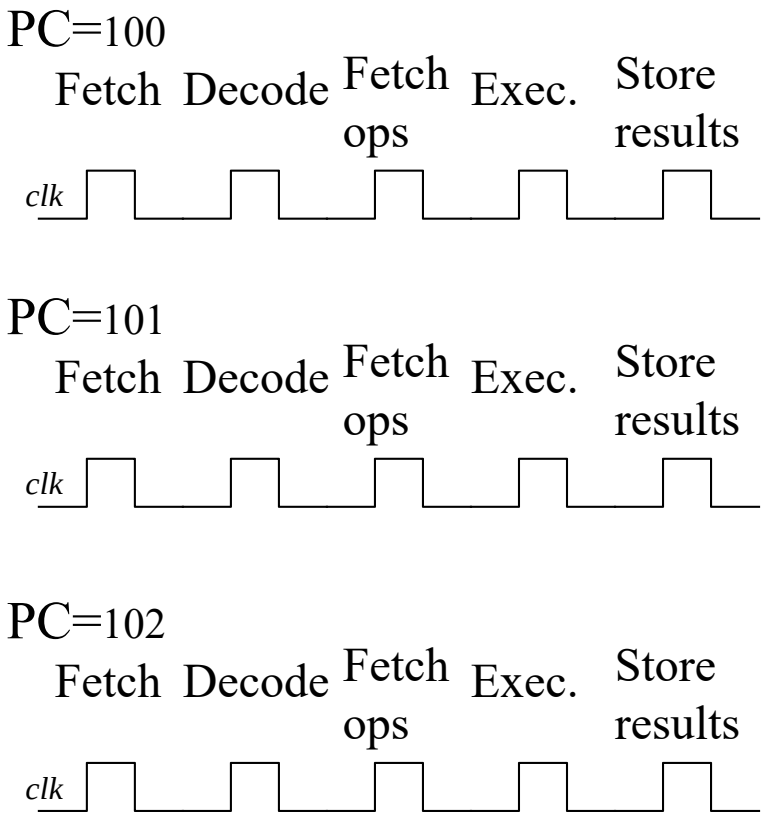


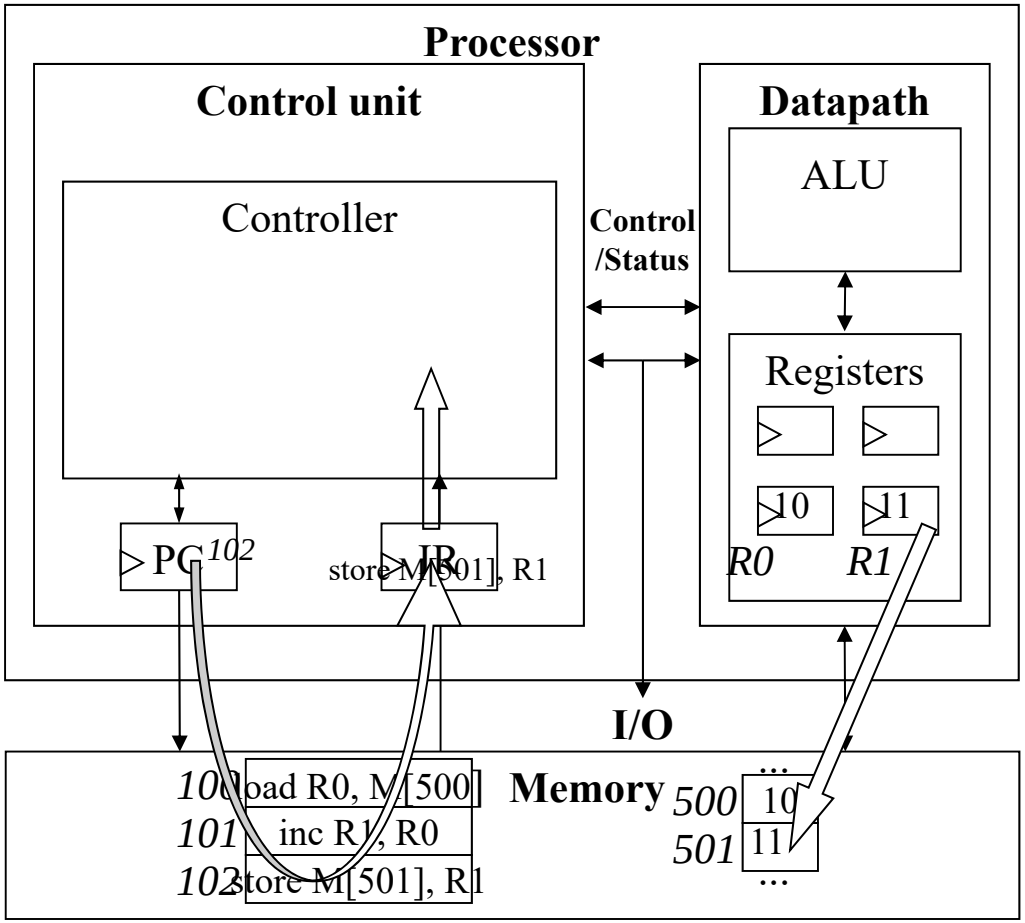
Instruction Cycles (II)





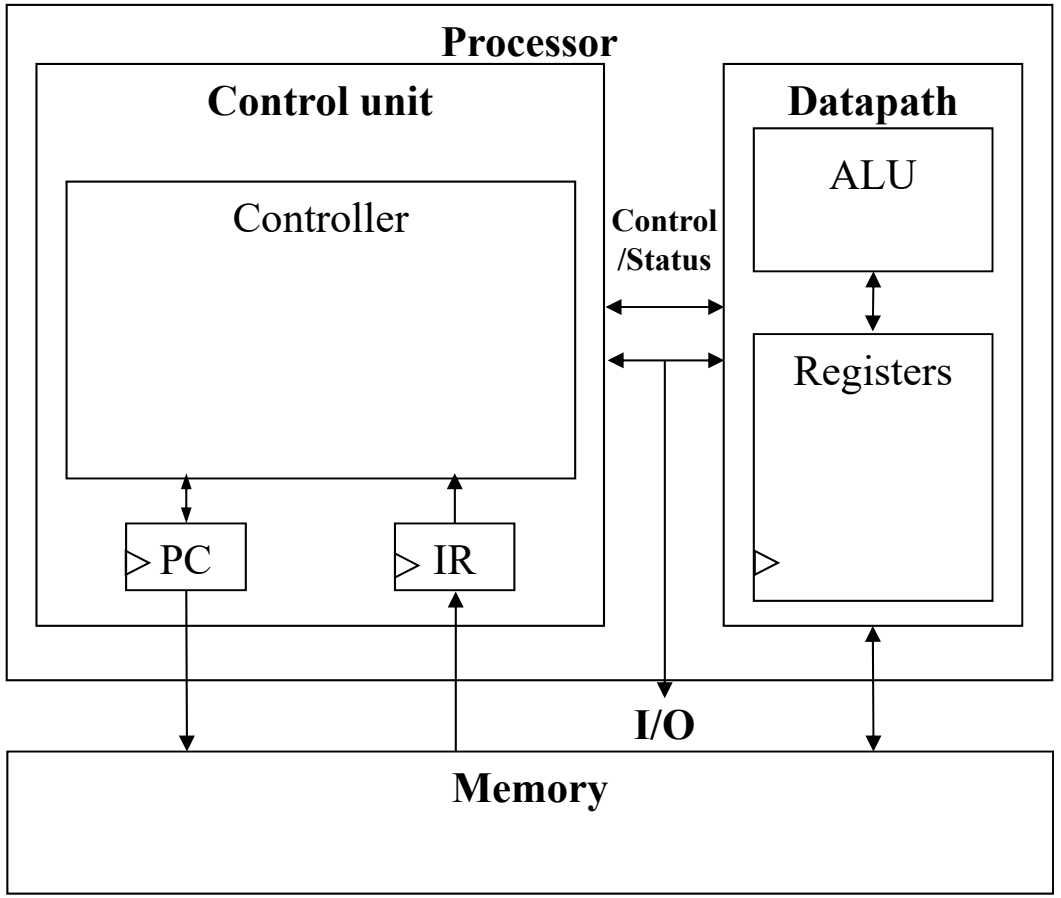
Instruction Cycles (III)





D. 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
- ***M-bit address***
  - PC size determines address space
  - M-bit PC
  - 2^M memory space
  - N x 2^M bits total



Architectural Considerations (II)

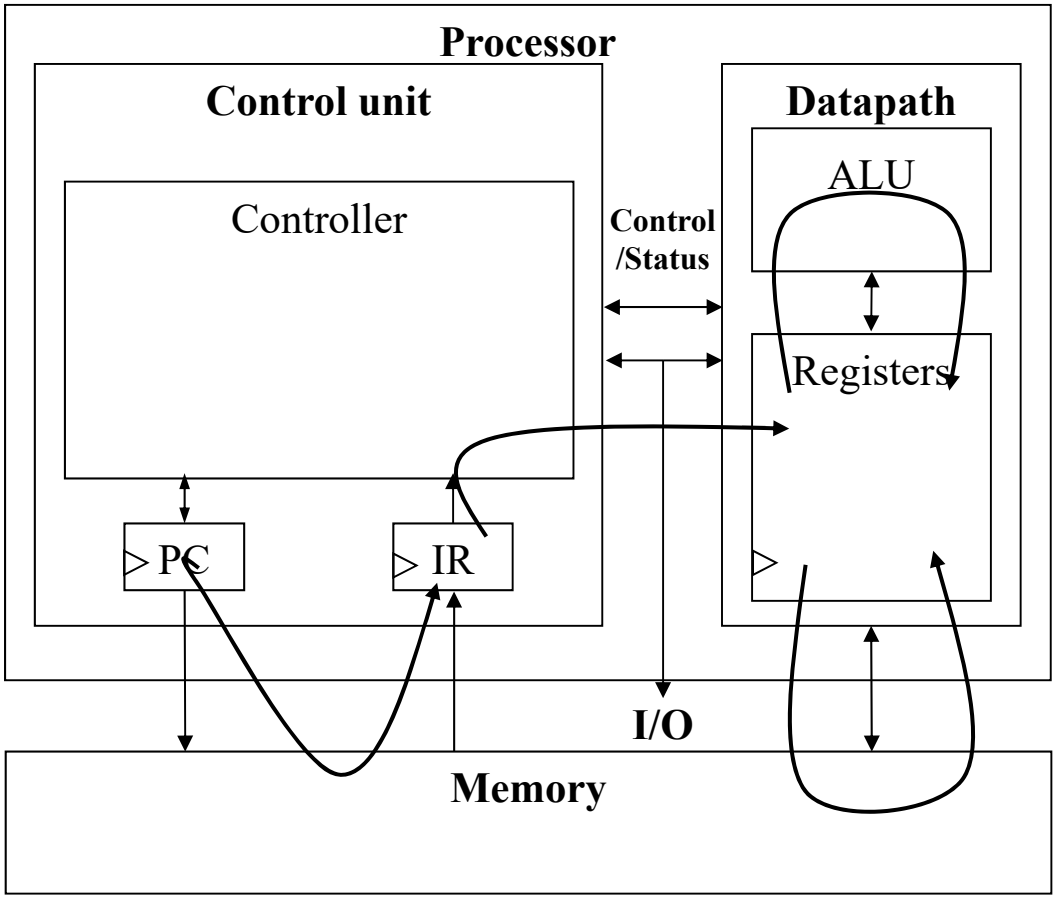
- **Sizes of data and address**

Microprocessor	Data	Address
8080 (1973)	8	16 (64 KB)
8086	16	20 (1 MB)
i386	32	32 (4 GB)
Pentium	32	32 (4 GB)
Pentium D	32	64 (16 EB)
Core 2	64	64 (16 EB)
Xscale	32	32 (4 GB)

- 1000^m:
  - 1 K kilo, 2 M mega, 3 G giga, 4 T tera, 5 P peta, 6 E exa, 7 Z zetta, 8 Y yotta

Architectural Considerations (III)

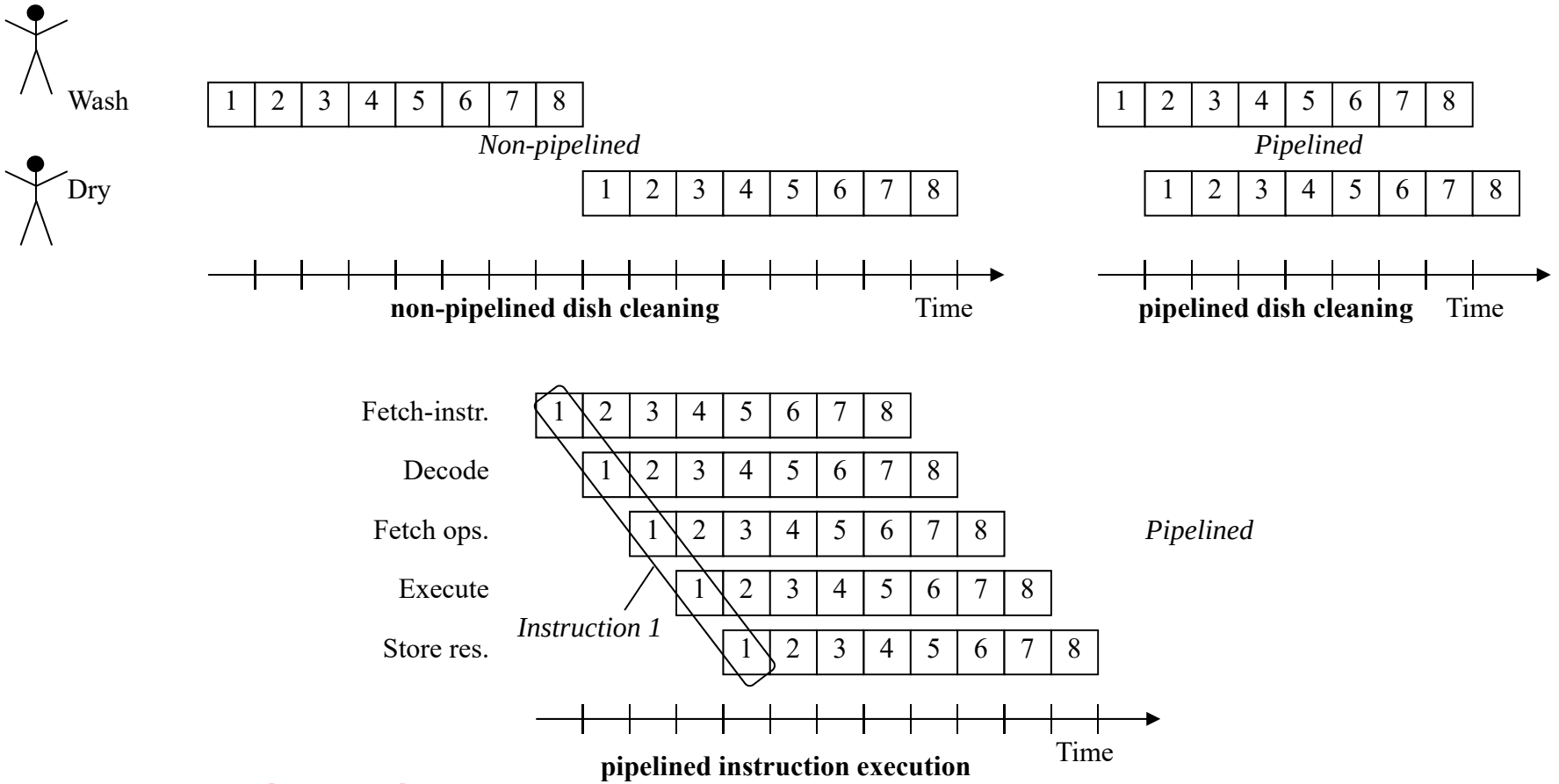
- **Clock frequency**
  - Inverse of clock period
  - *f (Hz)= 1/p(sec)*
  - *p* Must be *longer* than longest register to register delay in entire processor
  - *Memory access is often the longest.*



3.3 Operation

- **Instruction execution**
  - **1. Fetch instruction**
  - **2. Decode instruction**
  - **3. Fetch operands**
  - **4. Execute operation**
  - **5. Store results**
- Processor speed: 5 clock cycles / instruction
  - Processor throughput: 5N clock cycles / N instructions
- How can we improve the processor speed?
  - Fastest possible clock.
- How can we improve the processor throughput maximally?

Pipelining:
Increasing Instruction *Throughput*



- Branch problem
  - Sophisticated branch predictor

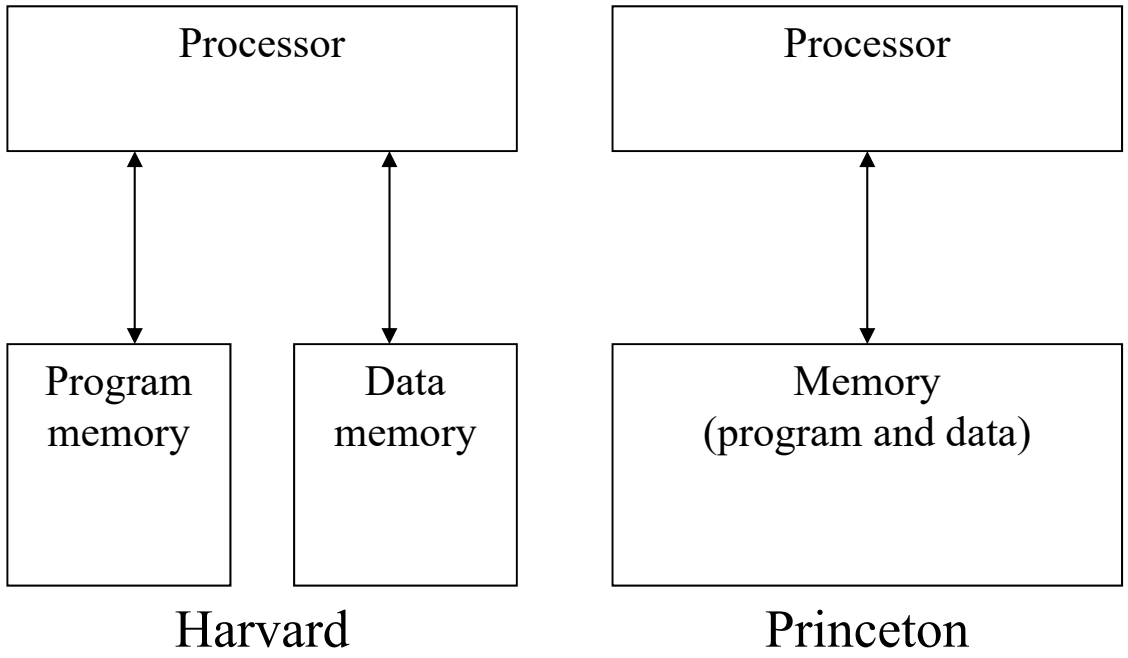
Superscalar and VLIW Architectures

- Performance can be improved by:
  - Faster clock (but there's a limit)
  - Pipelining: slice up instruction into stages, overlap stages
  - *Multiple ALUs* to support more than one instruction stream

- Superscalar
  - Scalar: non-vector operations
  - Fetches instructions in batches, executes as many as possible
    - May require extensive hardware to detect independent instructions
  - VLIW (Very Long Instruction Word): each word in memory has multiple independent instructions
    - Relies on the compiler to detect and schedule instructions
    - Currently growing in popularity.

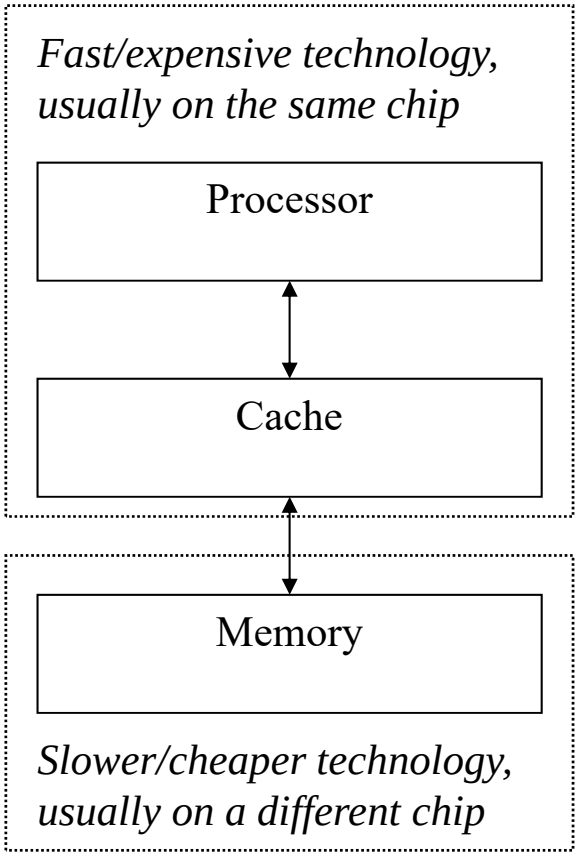
Two Memory Architectures

- **Princeton**
  - Fewer memory wires
- **Harvard**
  - Simultaneous program and data memory access



Cache Memory

- Memory access may be slow
- **Cache** is small but fast memory close to processor
  - Holds copy of part of memory
  - Hits and misses



Cache Memory (II)

- Cache vs. Register & Memory

Type	Impl.	Speed	Cost/bit
Register	D f/f	Fastest	High
Cache	SRAM	Faster	Medium
Memory	DRAM	Fast	Low

- PC
  - 8 regs. 1-2 MB cache. 1 – 4 GB memory.

Part 1/4 Summary

- General-purpose processors
  - Good performance, low NRE, flexible
- Controller, datapath, and memory
- Instruction execution
  - Fetch, decode, fetch operand, execute, and store
- Performance enhancement
  - Pipelining
  - Superscalar & VLIW

References

- [1] Frank Vahid, “Embedded system design: A unified hardware/software introduction”, John Wiley & Sons, 2002.