Computer Organization and Architecture

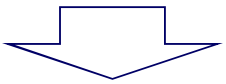
Top-Level View of Computer Function Interconnection

Program Concept

- **Hardwired systems are inflexible**
- **General purpose hardware can do different tasks, given correct control signals**
- **Instead of re-wiring, supply a new set of control signals**

What is a program?

- **A sequence of steps (instructions)**
- **For each step, an arithmetic or logical operation is done**
- **For each operation, a different set of control signals is needed**



Instruction

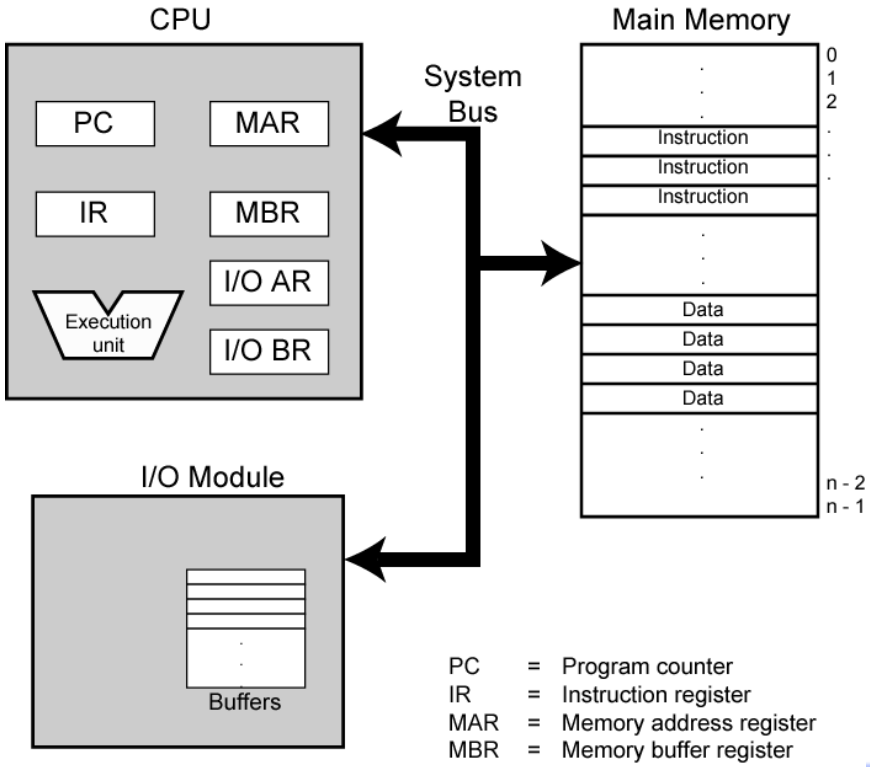
Function of Control Unit

- **For each operation a unique code is provided**
  - —e.g. ADD, MOVE
- **A hardware segment accepts the code and issues the control signals**
- **We have a computer!**

Components

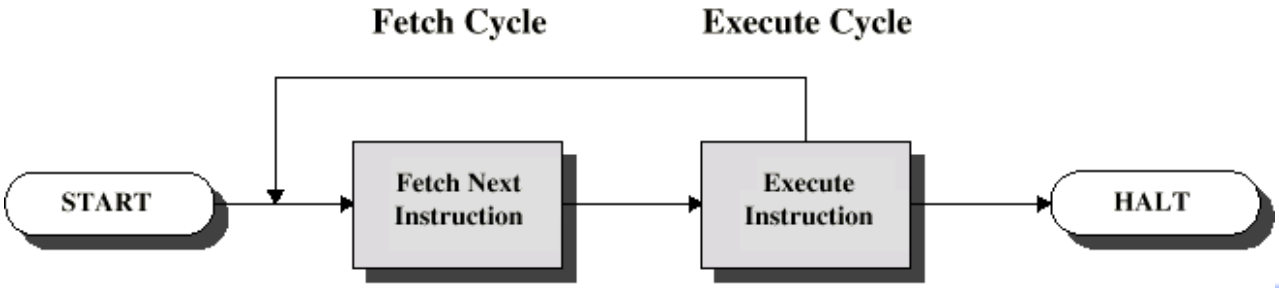
- **Central Processing Unit (CPU)**
  - —The Control Unit
  - —The Arithmetic and Logic Unit (ALU)
- **Input/output: Data and instructions need to get into the system and results out**
- **Main memory: Temporary storage of code and results is needed**

Computer Components: Top Level View



Instruction Cycle

- **Two steps:**
  - —Fetch
  - —Execute



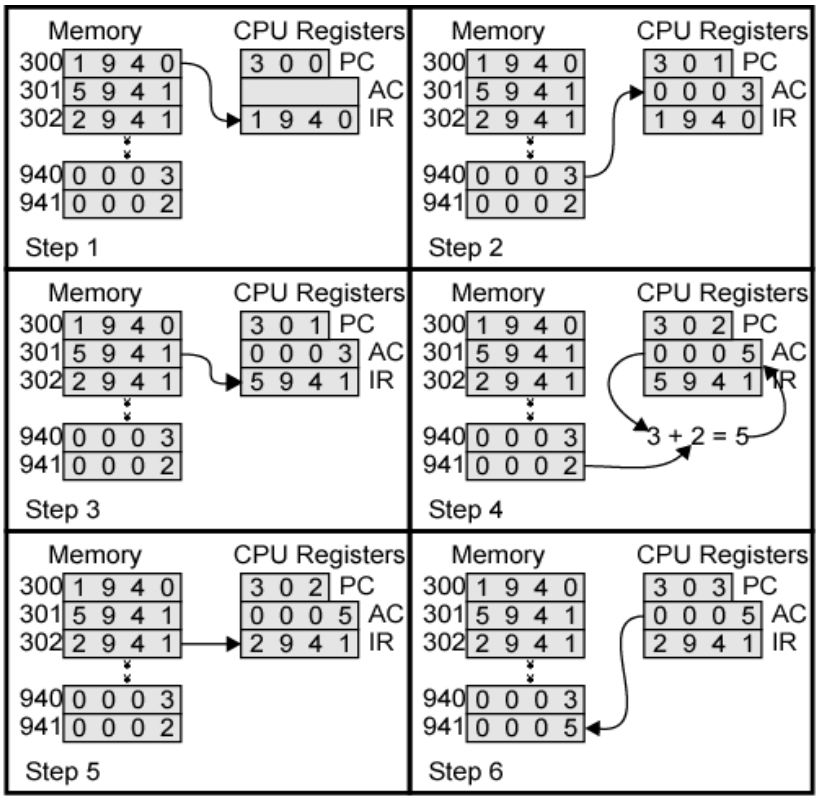
Fetch Cycle - Details

- **Program Counter (PC) holds address of next instruction to fetch**
- **Processor fetches instruction from memory location pointed to by PC**
- **Increment PC**
  - —Unless told otherwise
- **Instruction loaded into Instruction Register (IR)**
- **Processor interprets instruction and performs required actions**

Execute Cycle - Types

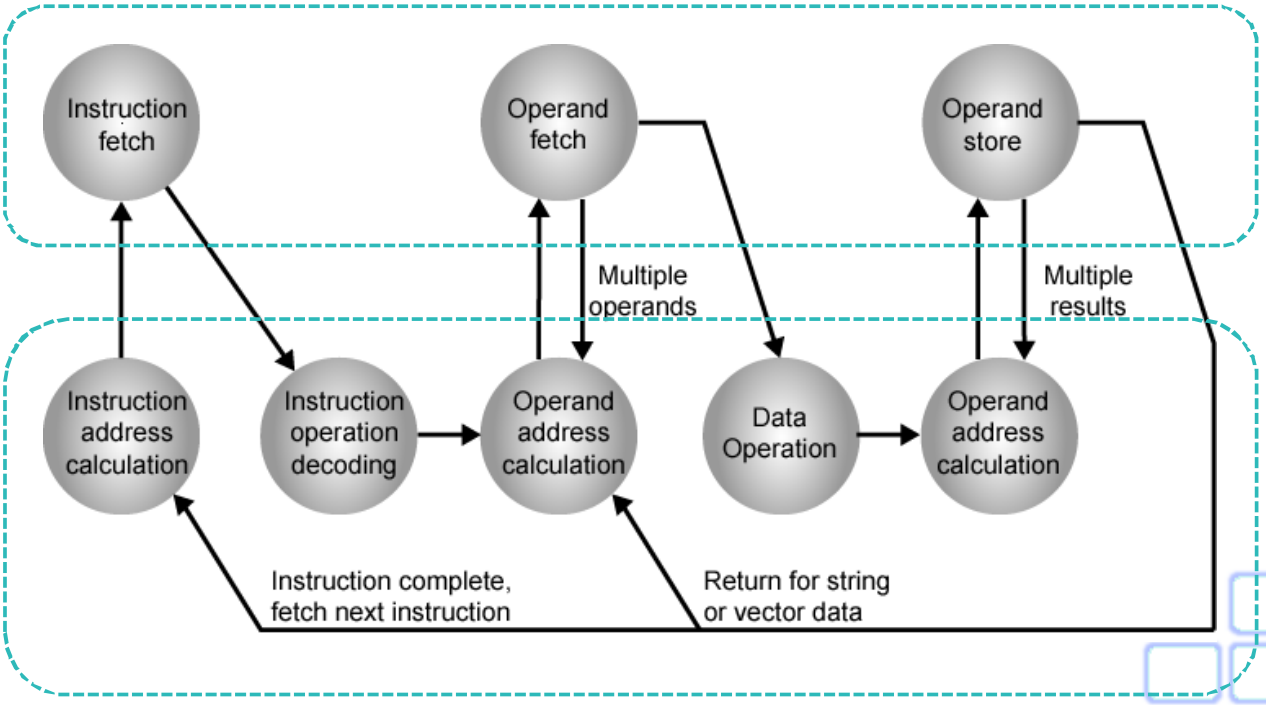
- **Processor-memory**
  - —Data transfer between CPU and main memory
- **Processor I/O**
  - —Data transfer between CPU and I/O module
- **Data processing**
  - —Some arithmetic or logical operation on data
- **Control**
  - —Change of sequence of operations
  - —e.g. jump
- **An instruction: Combination of above**

Example of Program Execution



940 <- 940+941

Instruction Cycle State Diagram



Interrupts

- **Mechanism by which other modules (e.g. I/O) may interrupt normal sequence of processing**
- **Improve processing efficiency**

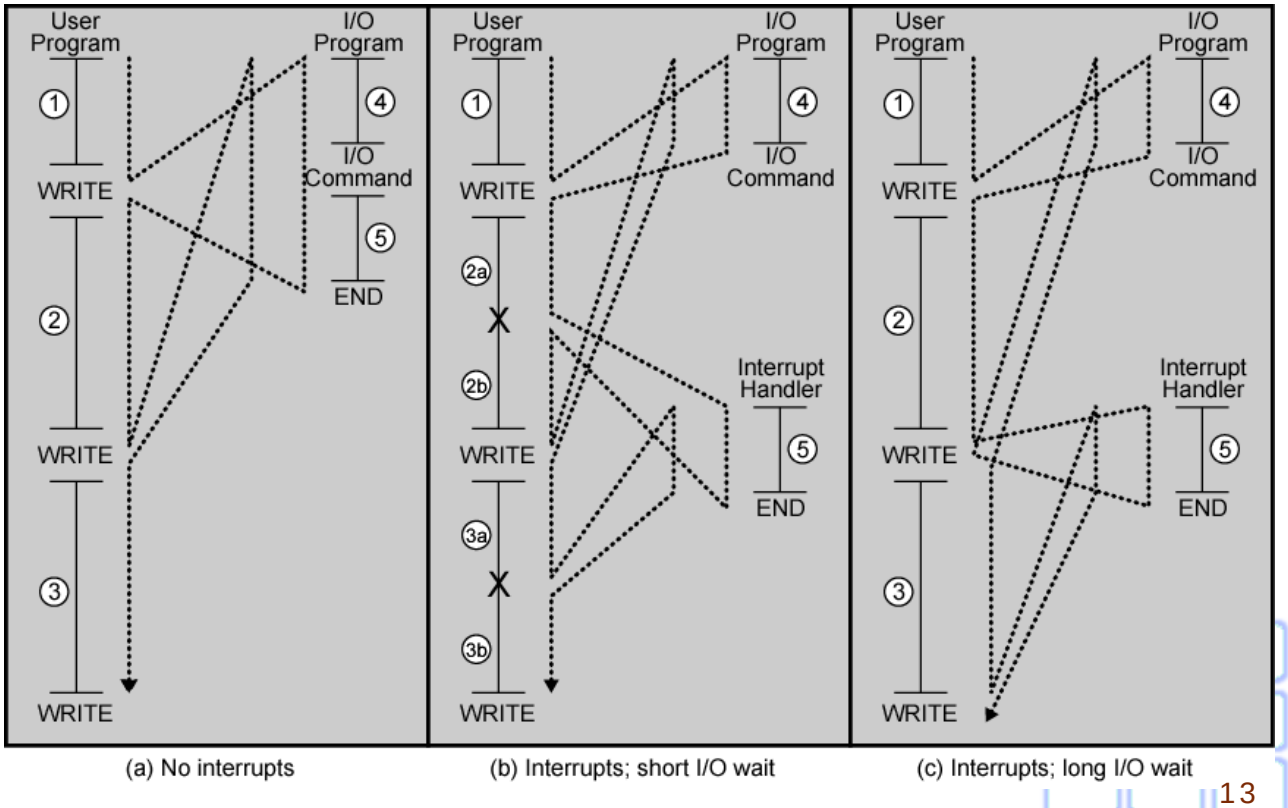
Program
e.g. overflow, division by zero

Timer
Generated by internal processor timer
Used in pre-emptive multi-tasking

I/O
from I/O controller

Hardware failure
e.g. memory parity error

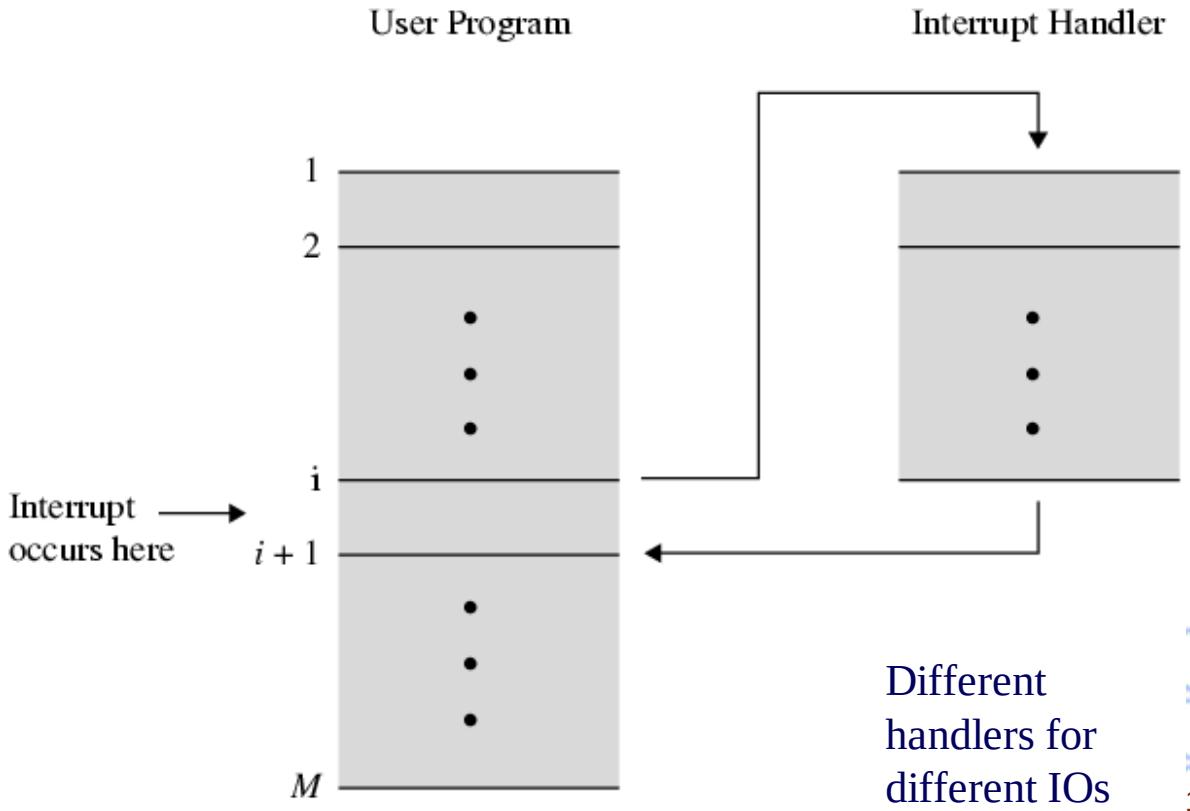
Program Flow Control



Interrupt Cycle

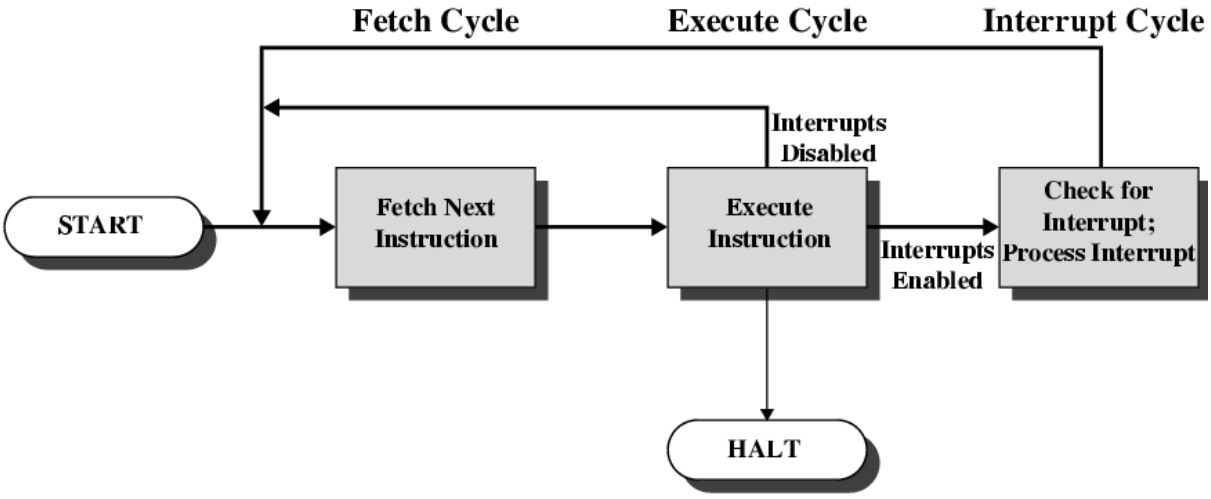
- **Added to instruction cycle**
- **Processor checks for interrupt**
  - —Indicated by an interrupt signal
- **If no interrupt, fetch next instruction**
- **If interrupt pending:**
  - —Suspend execution of current program
  - —Save context
  - —Set PC to start address of interrupt handler routine
  - —Process interrupt
  - —Restore context and continue interrupted program

Transfer of Control via Interrupts

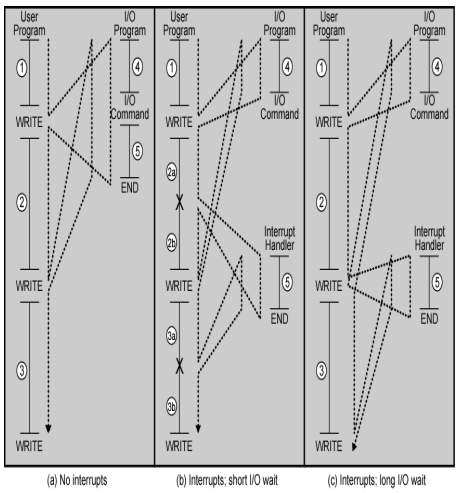


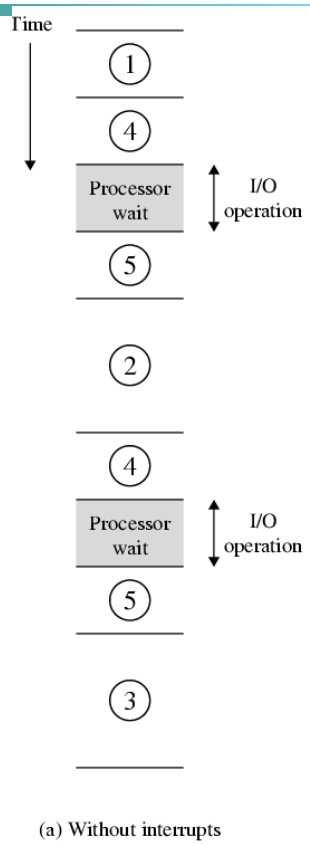
Different handlers for different IOs

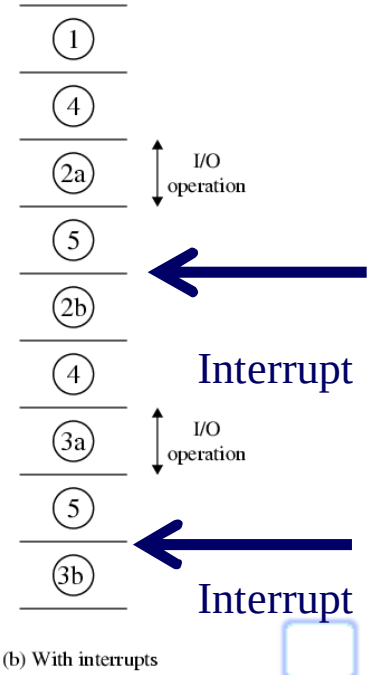
Instruction Cycle with Interrupts



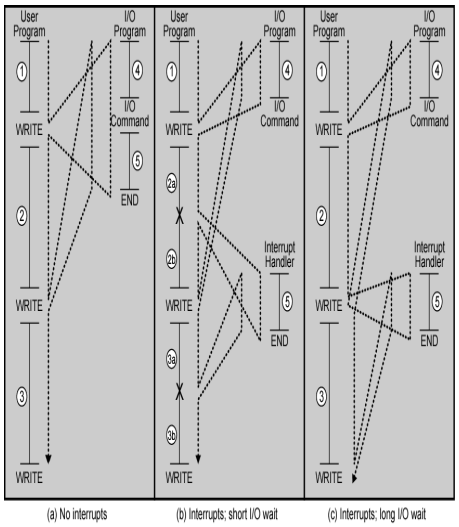
Program Timing
Short I/O Wait

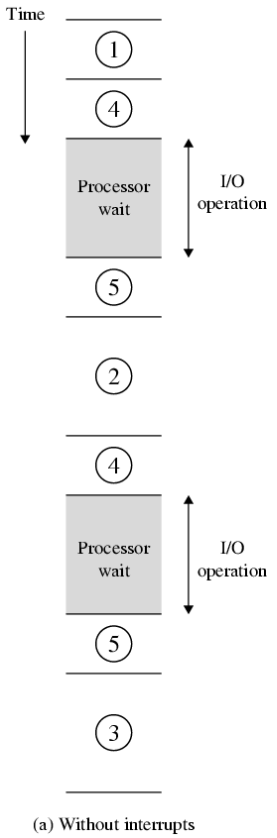


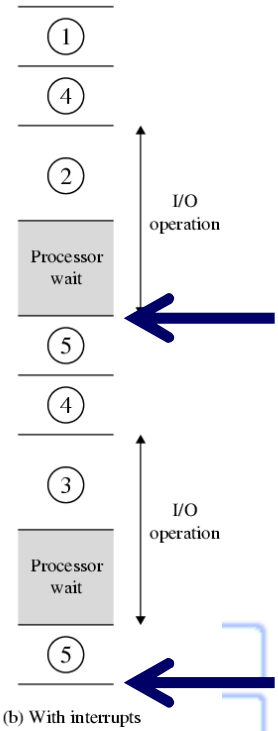




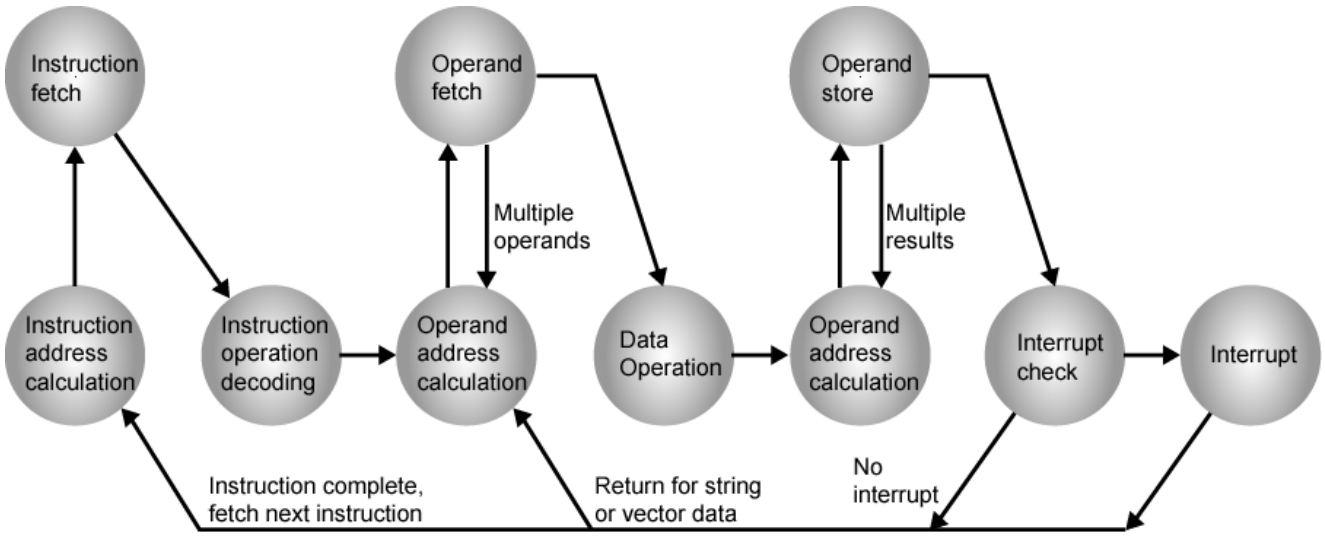
Program Timing
Long I/O Wait







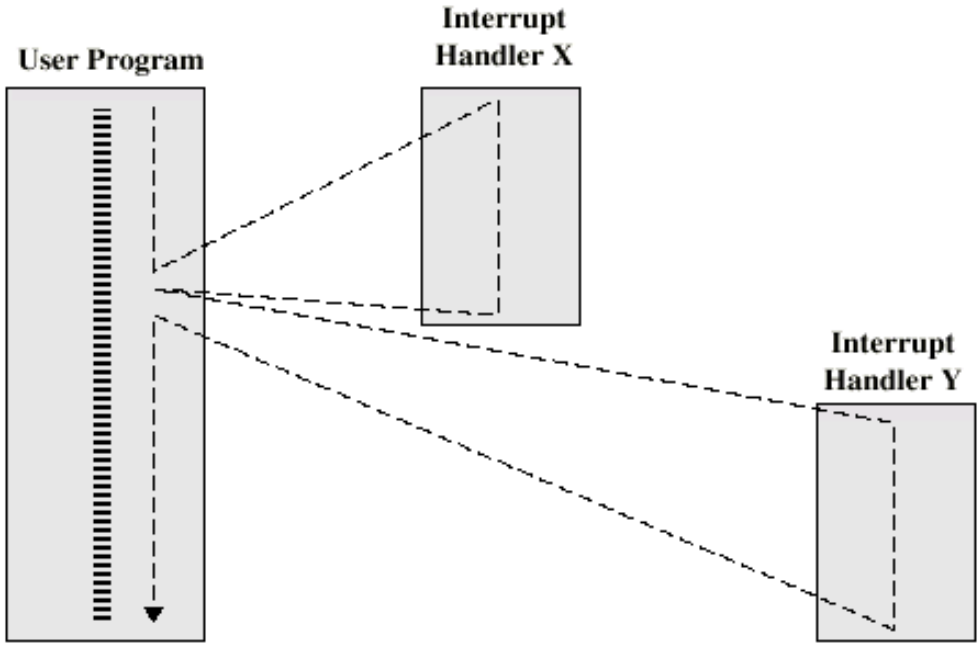
Instruction Cycle (with Interrupts) - State Diagram



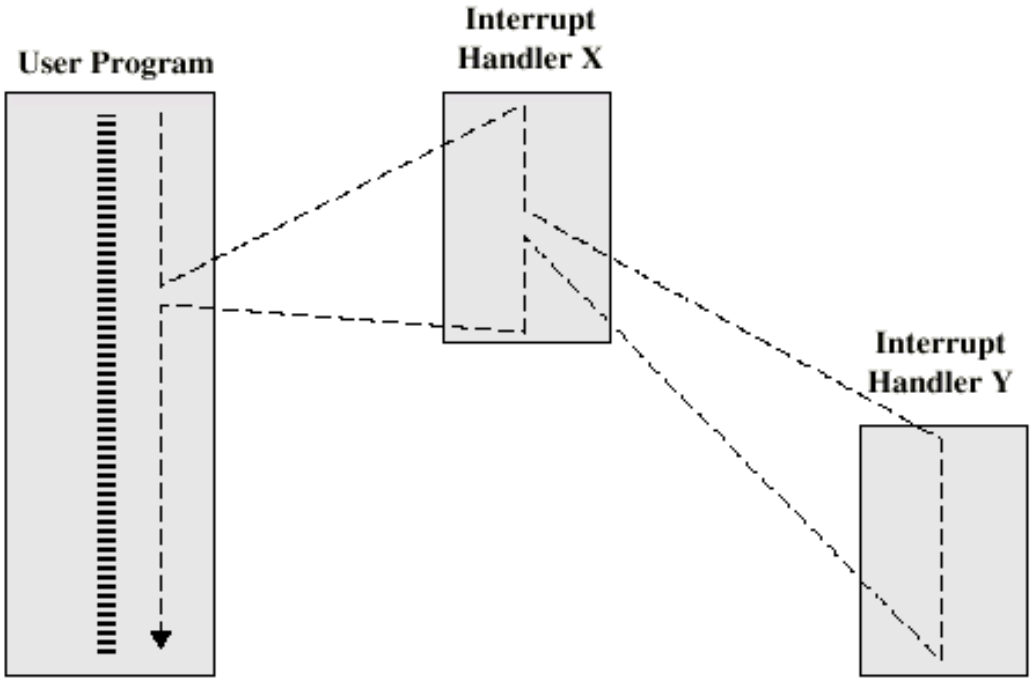
Multiple Interrupts

- **1) Disable interrupts**
  - —Processor will ignore further interrupts whilst processing one interrupt
  - —Interrupts remain pending and are checked after first interrupt has been processed
  - —Interrupts handled in sequence as they occur
- **2) Define priorities**
  - —Low priority interrupts can be interrupted by higher priority interrupts
  - —When higher priority interrupt has been processed, processor returns to previous interrupt

Multiple Interrupts - Sequential

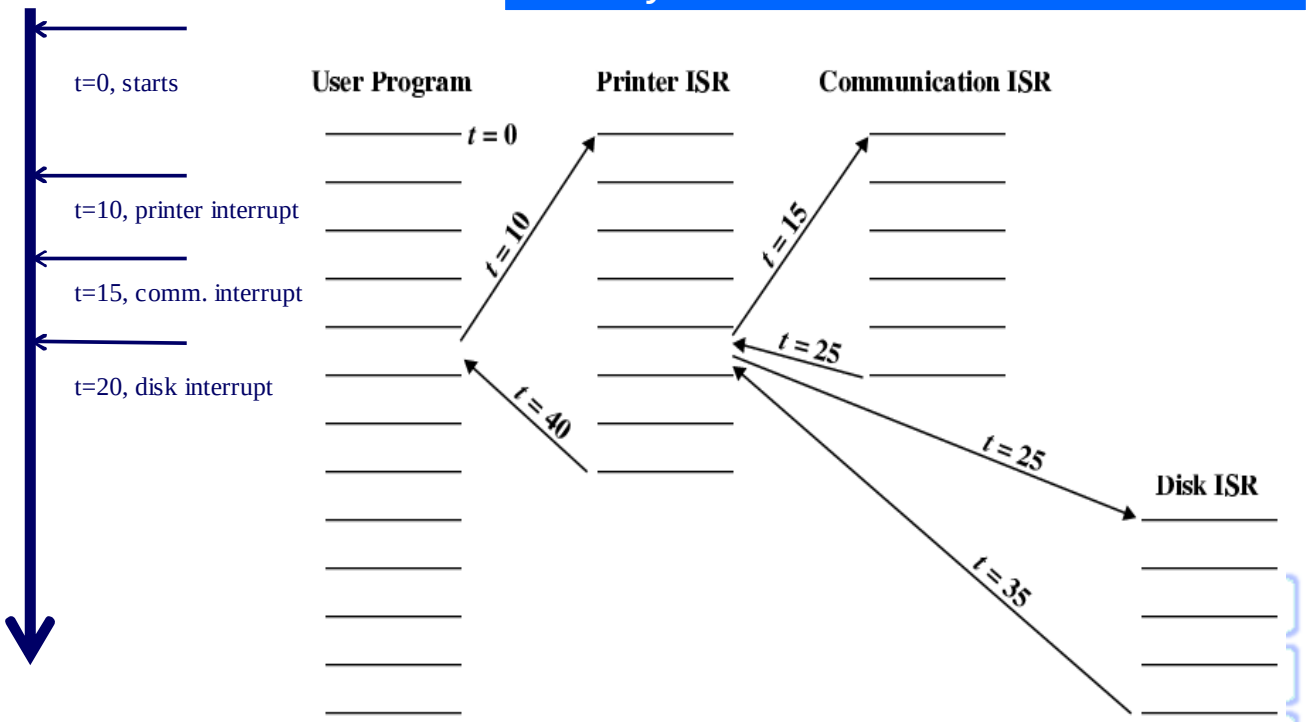


Multiple Interrupts – Nested



Time Sequence of Multiple Interrupts

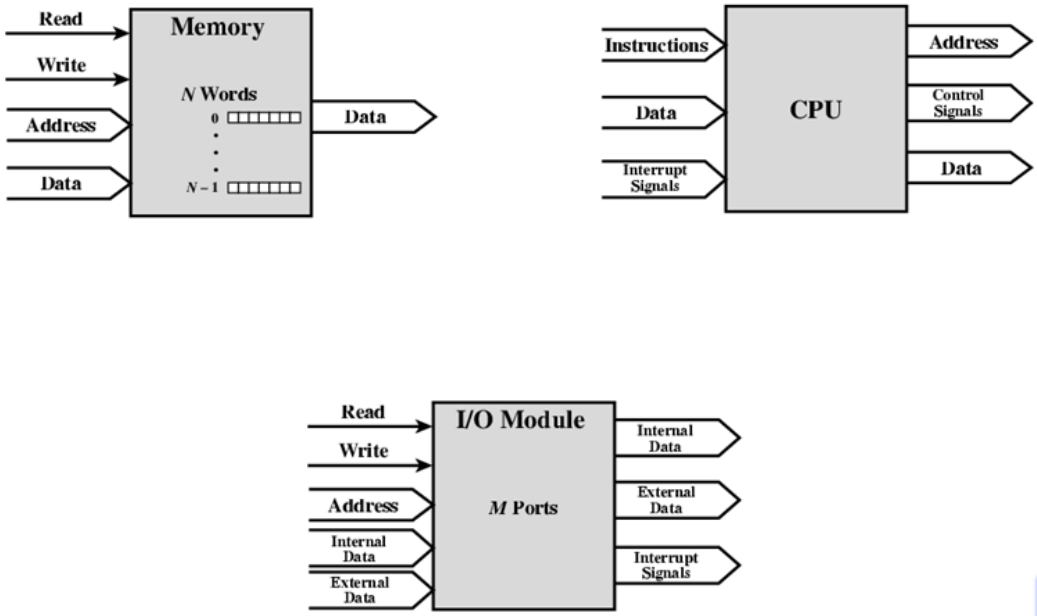
Priority: Printer < Disk < Comm.



Interconnection

- **All the units must be connected**
- **Different type of connection for different type of unit**
  - —Memory
  - —Input/Output
  - —CPU

Computer Modules



Memory Connection

- **Receives and sends data**
- **Receives addresses (of locations)**
- **Receives control signals**
  - —Read
  - —Write
  - —Timing

Input/Output Connection (1)

- **Similar to memory from computer's viewpoint**
- **Output**
  - —Receive data from computer
  - —Send data to peripheral
- **Input**
  - —Receive data from peripheral
  - —Send data to computer

Input/Output Connection (2)

- **Receive control signals from computer**
- **Send control signals to peripherals**
  - —e.g. spin disk
- **Receive addresses from computer**
  - —e.g. port number to identify peripheral
- **Send interrupt signals (control)**

CPU Connection

- **Reads instruction and data**
- **Writes out data (after processing)**
- **Sends control signals to other units**
- **Receives (& acts on) interrupts**

Interconnection - Buses

- **There are a number of possible interconnection systems**
- **Single and multiple BUS structures are most common**

e.g. **Unibus** (DEC-PDP)

```
18 A00-A17 - Address Lines
16 D00-D15 - Data Lines
 4 BR4-BR7 - Bus (Interrupt) Requests at priorities 4 (lowest) through 7 (highest)
 4 BG4-BG7 - Bus (Interrupt) Grants at priorities 4 (lowest) through 7 (highest)
 1 NPR     - Non Processor (DMA) Request
 1 NPG     - Non Processor (DMA) Grant
 1 ACLO    - AC Low
 1 DCLO    - DC Low
 1 MSYNC   - Master Sync
 1 SSYNC   - Slave Sync
 1 BBSY    - Bus Busy
 1 SACK    - Selection Acknowledge
 1 INIT    - Bus Init
 1 INTR    - Interrupt Request
 1 PA      - Parity control
 1 PB      - Parity control
 2 C0-C1   - Cyce Control Lines:
 2 +5v     - Power Lines (not counted as part of the 56)
14 Gnd     - Ground Lines (not counted as part of the 56)
```

What is a Bus?

A communication pathway connecting two or more devices

- **Usually broadcast**
- **Often grouped**
  - —A number of channels in one bus
  - —e.g. 32 bit data bus is 32 separate single bit channels
- **Power lines may not be shown**

Characteristics: Shared

Cannot transmit at the same time

Data Bus

- **Carries data**
  - —Remember that there is no difference between "data" and "instruction" at this level
- **Width is a key determinant of performance**
  - —8, 16, 32, 64 bit

Address bus

- **Identify the source or destination of data**
  - —e.g. CPU needs to read an instruction (data) from a given location in memory
- **Bus width determines maximum memory capacity of system**
  - —e.g. 8080 has 16 bit address bus giving 64k address space

Control Bus

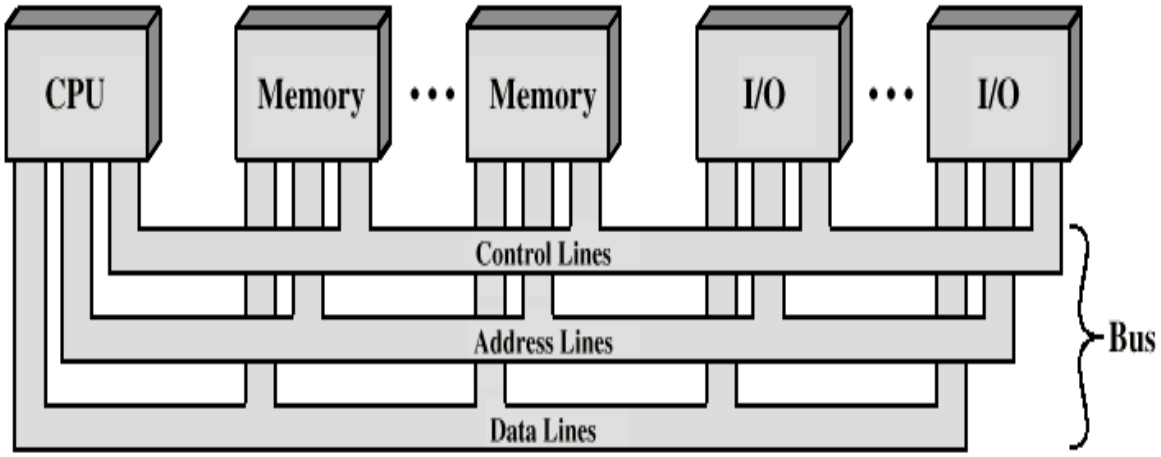
Control and timing information

- —Memory read/write signal
- —Interrupt request
- —Clock signals

minibus

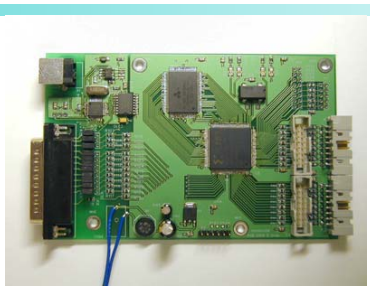
```
18 A00-A17 - Address Lines
16 D00-D15 - Data Lines
 4 BR4-BR7 - Bus (Interrupt) Requests at priorities 4 (lowest) through 7 (highest)
 4 BG4-BG7 - Bus (Interrupt) Grants at priorities 4 (lowest) through 7 (highest)
 1 NPR     - Non Processor (DMA) Request
 1 NPG     - Non Processor (DMA) Grant
 1 ACLO    - AC Low
 1 DCLO    - DC Low
 1 MSYNC   - Master Sync
 1 SSYNC   - Slave Sync
 1 BBSY    - Bus Busy
 1 SACK    - Selection Acknowledge
 1 INIT    - Bus Init
 1 INTR    - Interrupt Request
 1 PA      - Parity control
 1 PB      - Parity control
 2 C0-C1   - Cyce Control Lines:
 2 +5v     - Power Lines (not counted as part of the 56)
14 Gnd     - Ground Lines (not counted as part of the 56)
```

Bus Interconnection Scheme

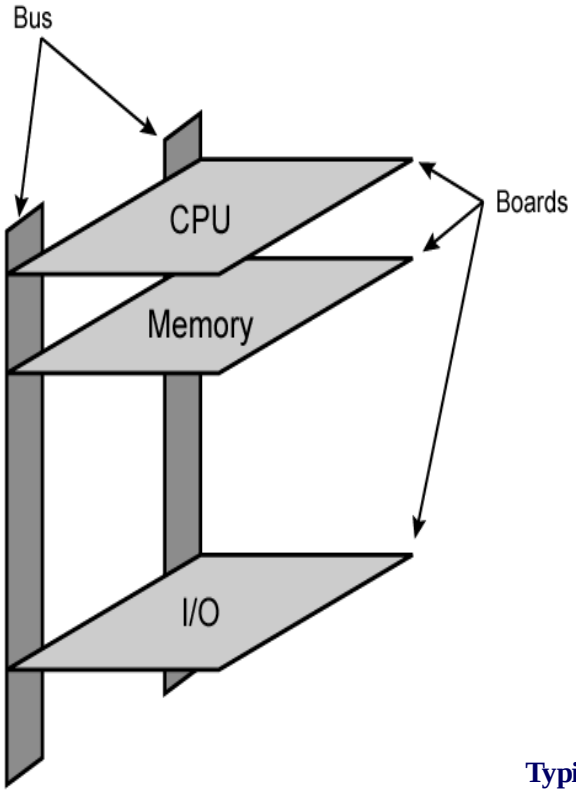


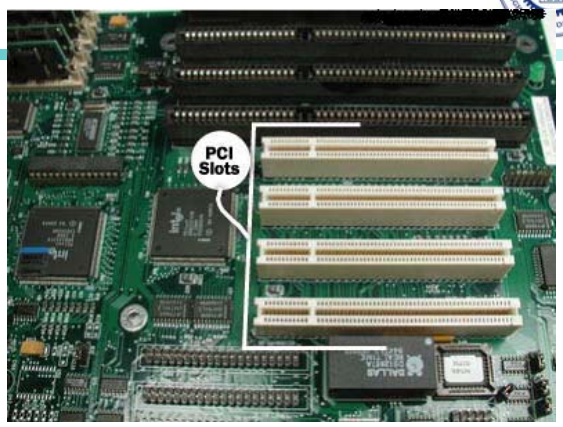
What do buses look like?

- **Parallel lines on circuit boards**
- **Ribbon cables**
- **Strip connectors on mother boards**
  - —e.g. PCI
- **Sets of wires /electrical lines**



Physical Realization of Bus Architecture





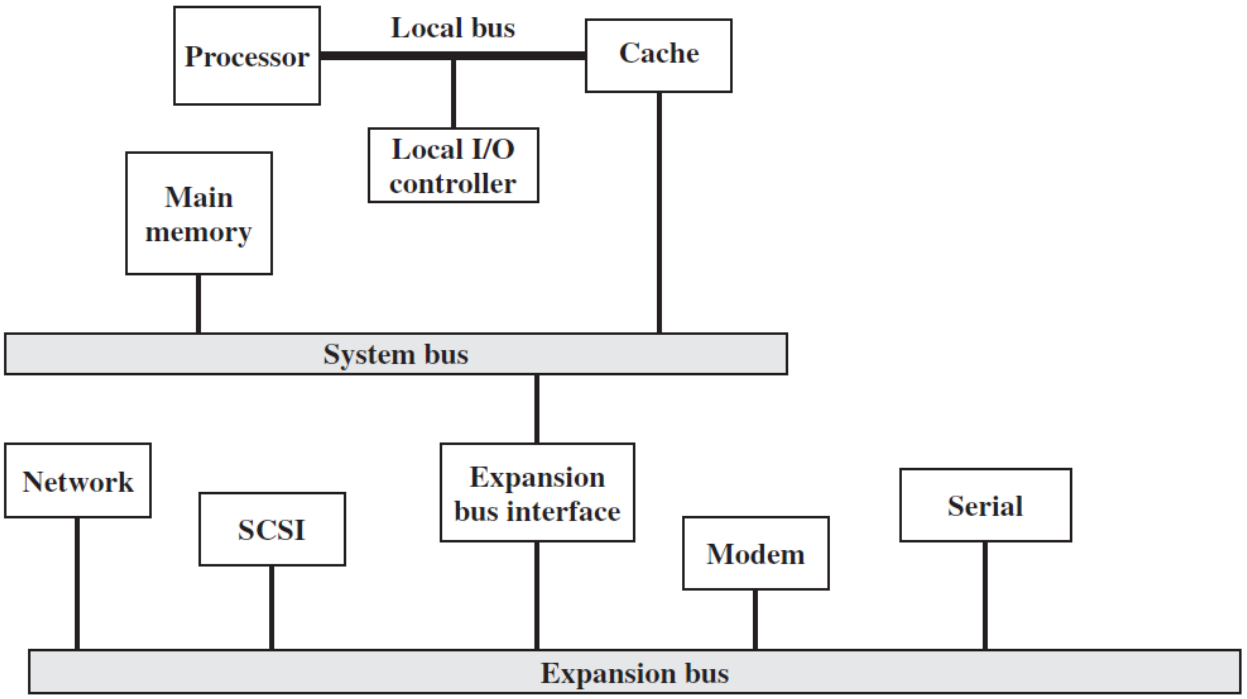
PCI slots on a motherboard

Typical example of an AGP-based graphics card

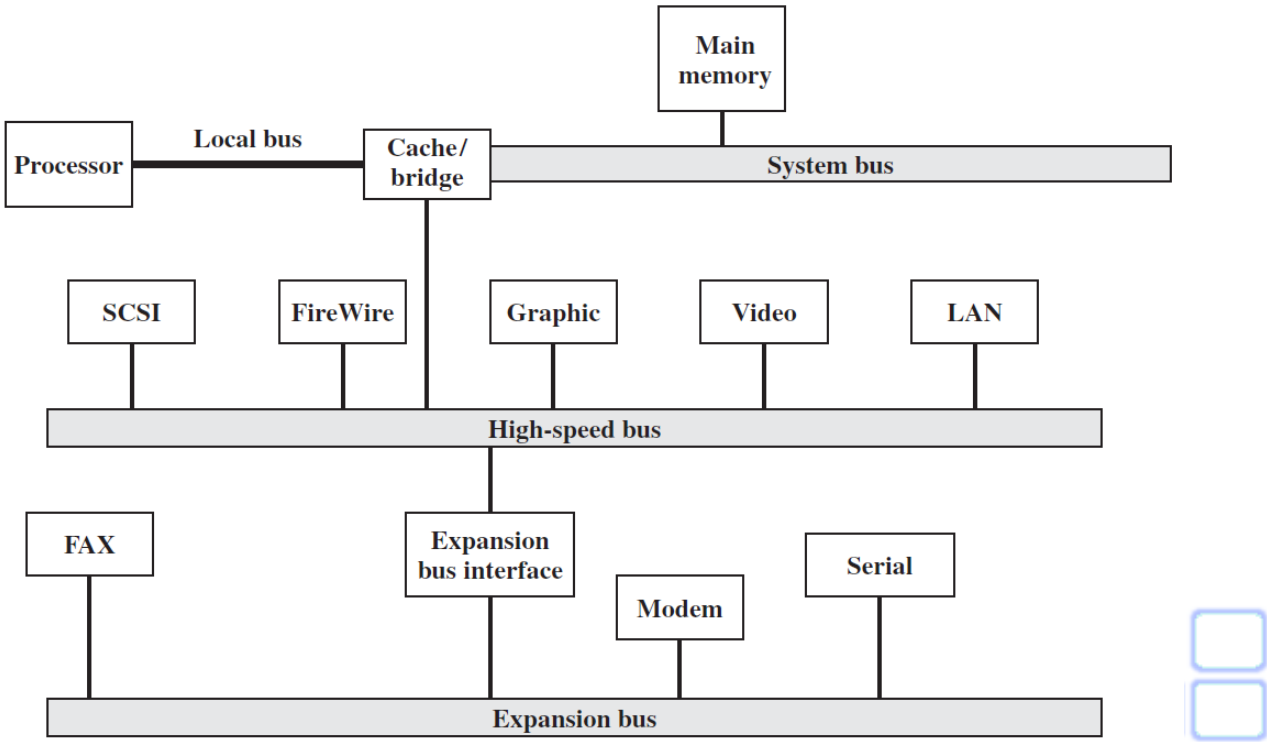
Single Bus Problems

- **Lots of devices on one bus leads to low performance**
  - —**Propagation delays**: Long data paths mean that co-ordination of bus use can adversely affect performance
  - —If **aggregate data** transfer approaches bus capacity
  - —Devices have very **different speeds**
- **Most systems use multiple buses to overcome these problems**

Traditional (ISA) (with cache)



High Performance Bus



Bus Design

Type	**Bus Width**
Dedicated	Address
Multiplexed	Data
Method of Arbitration	**Data Transfer Type**
Centralized	Read
Distributed	Write
Timing	Read-modify-write
Synchronous	Read-after-write
Asynchronous	Block

Bus Types

- **Dedicated**
  - —Separate data & address lines
- **Multiplexed**
  - —Shared lines
  - —Address valid or data valid control line
  - —**Advantage** - fewer lines
  - —**Disadvantages**
    - – More complex control
    - – Ultimate performance

Bus Arbitration

- **More than one module controlling the bus**
  - —e.g. CPU and DMA controller
- **Only one module may control bus at one time**
- **Arbitration may be centralised or distributed**

Centralised or Distributed Arbitration

- **Centralised**
  - —Single hardware device controlling bus access
    - – Bus Controller
    - – Arbiter
  - —May be part of CPU or separate
- **Distributed**
  - —Each module may claim the bus
  - —Control logic on all modules

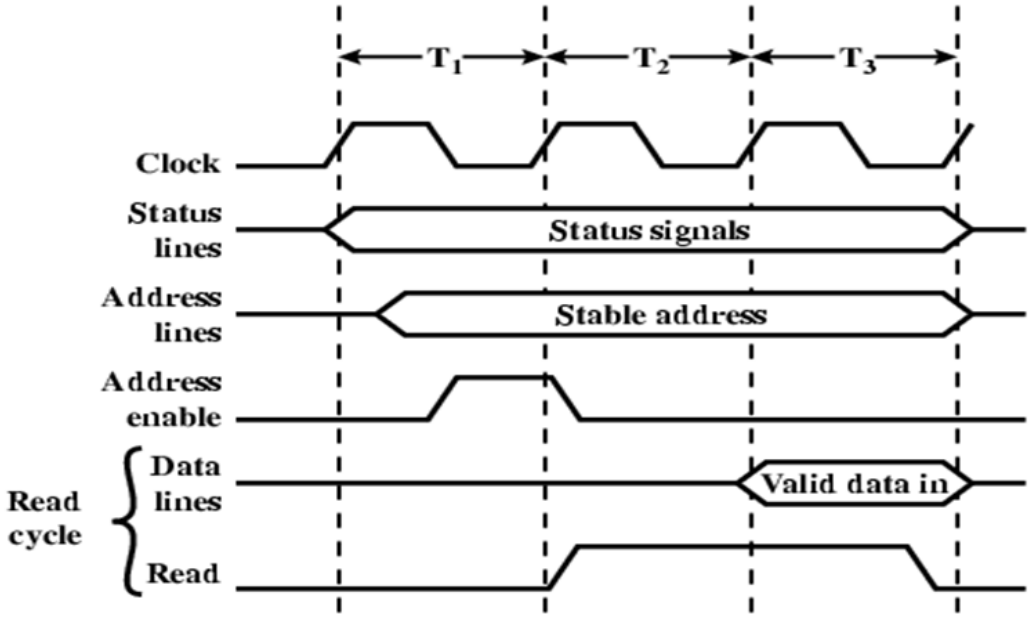
Timing

The way in which events are coordinated on the bus

All devices can read clock line

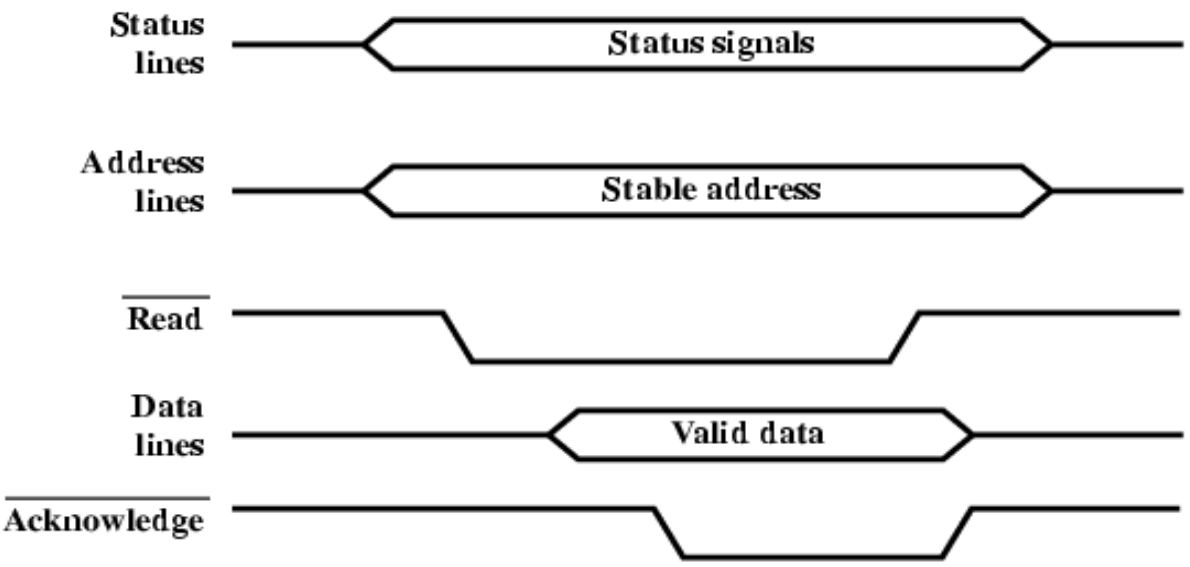
- **Synchronous**
  - —Events determined by clock signals
  - —Control Bus includes clock line
  - —A single 1-0 is a bus cycle
  - —Usually sync on leading edge
  - —Usually a single cycle for an event

Synchronous Timing Diagram



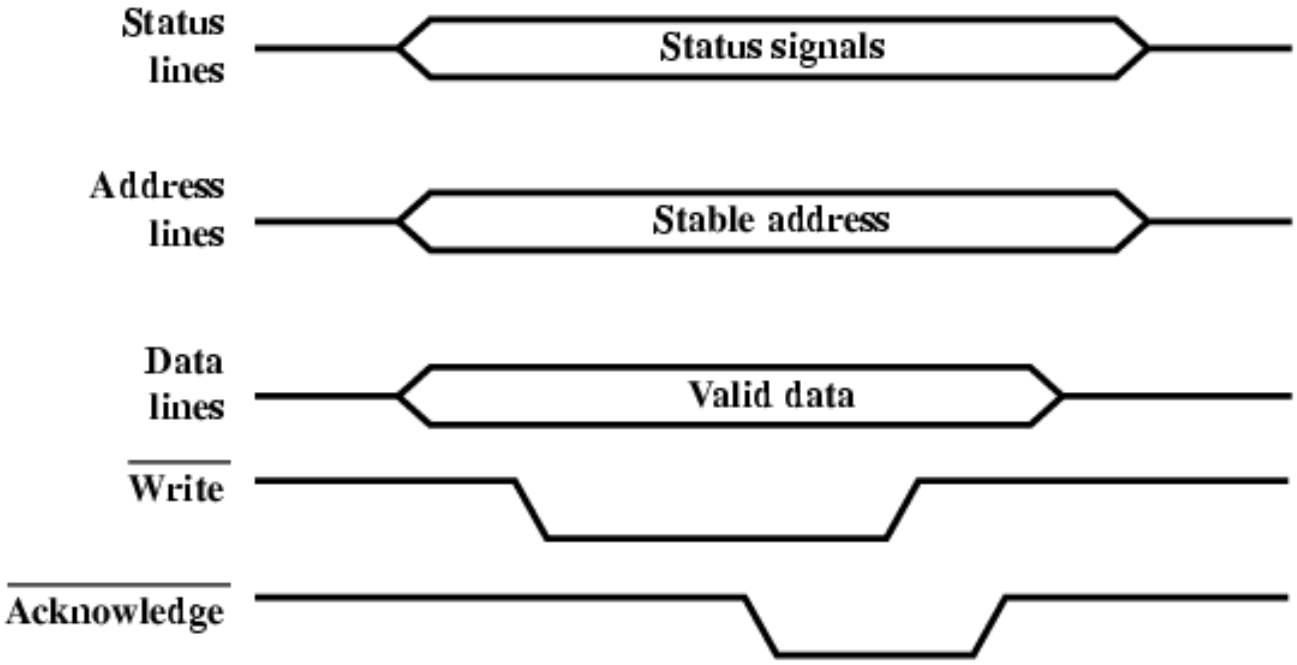
The occurrence of events on the bus is determined by a clock

Asynchronous Timing – Read Diagram



The occurrence of an events depends on a previous one.

Asynchronous Timing – Write Diagram



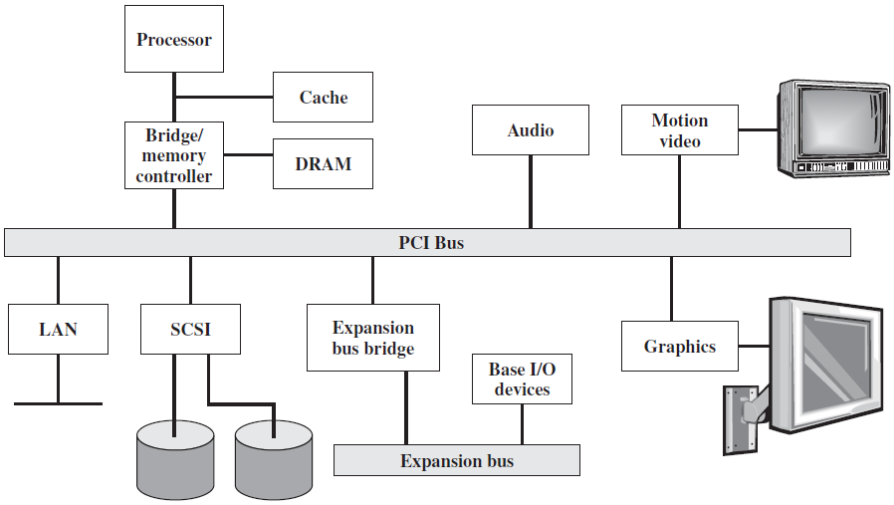
PCI Bus

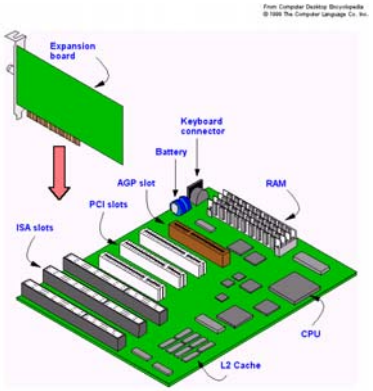
- **Peripheral Component Interconnection**

- **1990**
- **Intel released to public domain**
- **64 bit @ 66Mhz**
- **50 lines**

- Synchronous timing
- Centralized arbitration

PCI Structure





PCI Bus Lines (required)

- **Systems lines**
  - —Including clock and reset
- **Address & Data**
  - —32 time multiplex lines for address/data
  - —Interrupt & validate lines
- **Arbitration**
  - —Not shared
  - —Direct connection to PCI bus arbiter
- **Error lines**
- **Interface Control**

Designation	Type	Description
		System Pins
CLK	in	Provides timing for all transactions and is sampled by all inputs on the rising edge. Clock rates up to 33 MHz are supported.
RST#	in	Forces all PCI-specific registers, sequencers, and signals to an initialized state.
		Address and Data Pins
AD[31::0]	t/s	Multiplexed lines used for address and data
C/BE[3::0]#	t/s	Multiplexed bus command and byte enable signals. During the data phase, the lines indicate which of the four byte lanes carry meaningful data.
PAR	t/s	Provides even parity across AD and C/BE lines one clock cycle later. The master drives PAR for address and write data phases; the target drive PAR for read data phases.
		Interface Control Pins
FRAME#	s/t/s	Driven by current master to indicate the start and duration of a transaction. It is asserted at the start and deasserted when the initiator is ready to begin the final data phase.
IRDY#	s/t/s	Initiator Ready. Driven by current bus master (initiator of transaction). During a read, indicates that the master is prepared to accept data; during a write, indicates that valid data are present on AD.
TRDY#	s/t/s	Target Ready. Driven by the target (selected device). During a read, indicates that valid data are present on AD; during a write, indicates that target is ready to accept data.
STOP#	s/t/s	Indicates that current target wishes the initiator to stop the current transaction.
IDSEL	in	Initialization Device Select. Used as a chip select during configuration read and write transactions.
DEVSEL#	in	Device Select. Asserted by target when it has recognized its address. Indicates to current initiator whether any device has been selected.
		Arbitration Pins
REQ#	t/s	Indicates to the arbiter that this device requires use of the bus. This is a device-specific point-to-point line.
GNT#	t/s	Indicates to the device that the arbiter has granted bus access. This is a device-specific point-to-point line.
		Error Reporting Pins
PERR#	s/t/s	Parity Error. Indicates a data parity error is detected by a target during a write data phase or by an initiator during a read data phase.
SERR#	o/d	System Error. May be pulsed by any device to report address parity errors and critical errors other than parity.

PCI Bus Lines (Optional)

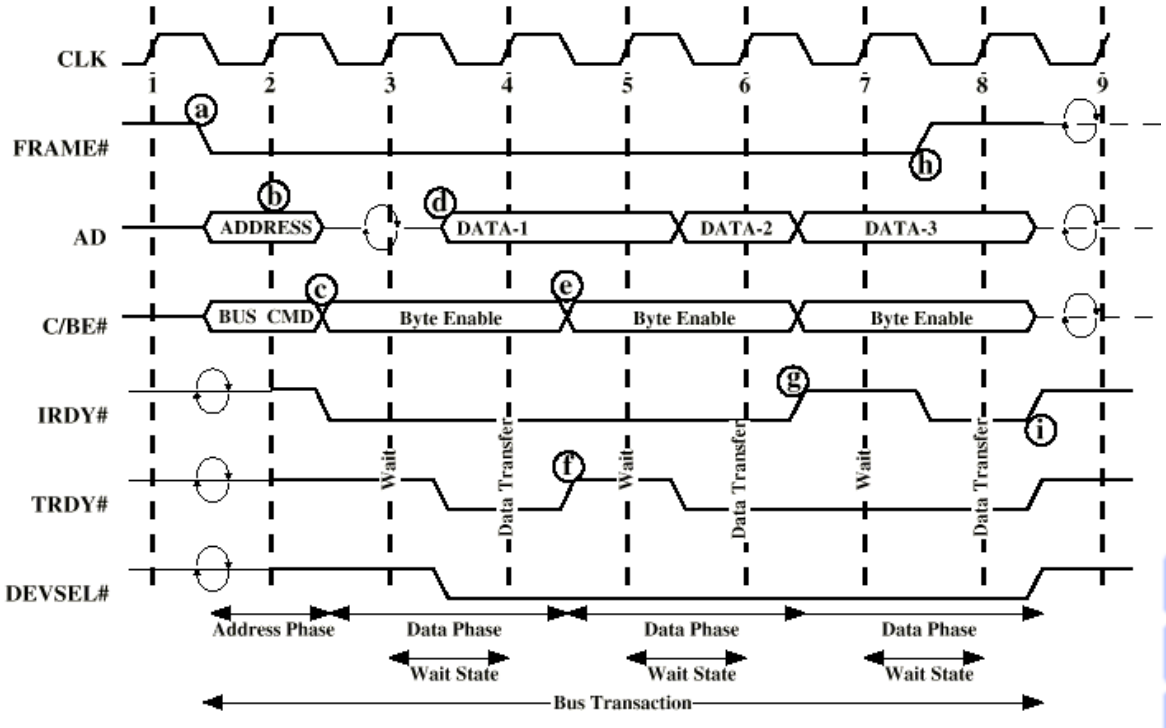
- **Interrupt lines**
  - —Not shared
- **Cache support**
- **64-bit Bus Extension**
  - —Additional 32 lines
  - —Time multiplexed
  - —2 lines to enable devices to agree to use 64-bit transfer
- **JTAG/Boundary Scan**
  - —For testing procedures

Designation	Type	Description
Interrupt Pins		
INTA#	o/d	Used to request an interrupt.
INTB#	o/d	Used to request an interrupt; only has meaning on a multifunction device.
INTC#	o/d	Used to request an interrupt; only has meaning on a multifunction device.
INTD#	o/d	Used to request an interrupt; only has meaning on a multifunction device.
Cache Support Pins		
SBO#	in/out	Snoop Backoff. Indicates a hit to a modified line.
SDONE	in/out	Snoop Done. Indicates the status of the snoop for the current access. Asserted when snoop has been completed.
64-Bit Bus Extension Pins		
AD[63::32]	t/s	Multiplexed lines used for address and data to extend bus to 64 bits.
C/BE[7::4]#	t/s	Multiplexed bus command and byte enable signals. During the address phase, the lines provide additional bus commands. During the data phase, the lines indicate which of the four extended byte lanes carry meaningful data.
REQ64#	s/t/s	Used to request 64-bit transfer.
ACK64#	s/t/s	Indicates target is willing to perform 64-bit transfer.
PAR64	t/s	Provides even parity across extended AD and C/BE lines one clock cycle later.
JTAG/Boundary Scan Pins		
TCK	in	Test clock. Used to clock state information and test data into and out of the device during boundary scan.
TDI	in	Test input. Used to serially shift test data and instructions into the device.
TDO	out	Test output. Used to serially shift test data and instructions out of the device.
TMS	in	Test mode Select. Used to control state of test access port controller.
TRST#	in	Test reset. Used to initialize test access port controller.

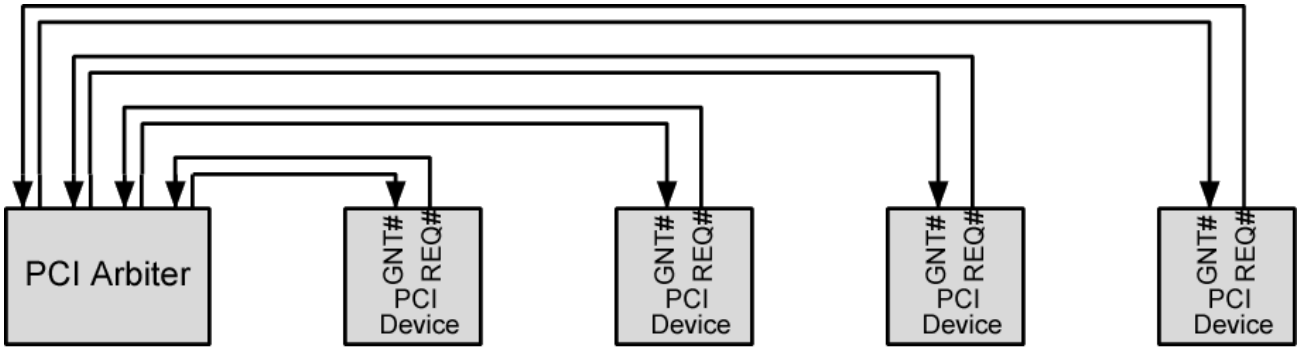
PCI Commands

- **Transaction between initiator (master) and target**
- **Master claims bus**
- **Determine type of transaction**
  - —e.g. I/O read/write
- **Address phase**
- **One or more data phases**

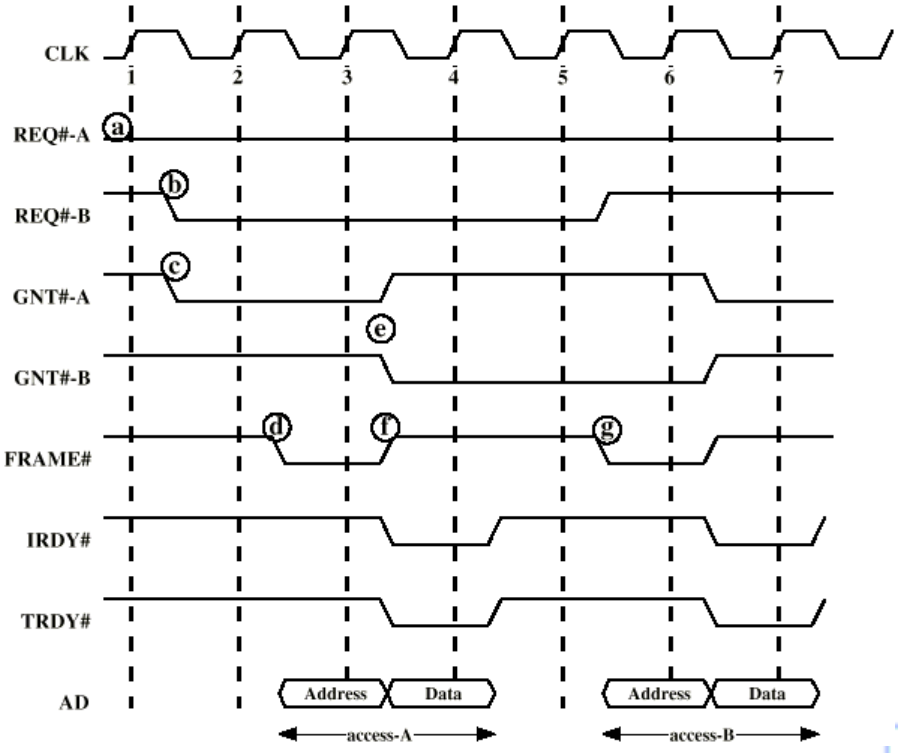
PCI Read Timing Diagram



PCI Bus Arbiter



PCI Bus Arbitration



Acknowledgements

- These slides contain material developed and copyright by:
  - —Arvind (MIT)
  - —Krste Asanovic (MIT/UCB)
  - —John Kubiatowicz (UCB)
  - —David Patterson (UCB)
  - —Geng Wang (SJTU)
  - —Yanmin Zhu (SJTU