Program Concept

• The process of connecting the various components in the desired configuration as a form of programming. The resulting “program” is in the form of hardware and is termed a hardwired program.

• 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.
- For each step, an arithmetic or logical operation is done.
- For each operation, a different set of control signals is needed.
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.

• The basic function performed by a computer is execution of a program, which consists of a set of instructions stored in memory.

• The processor does the actual work by executing instructions specified in the program.
Components

- The Control Unit and the Arithmetic and Logic Unit constitute the Central Processing Unit.
- Data and instructions need to get into the system and results out.
  - Input/output.
- Temporary storage of code and results is needed.
  - Main memory.
Computer Components: Top Level View

CPU
- PC
- IR
- MAR
- MBR
- I/O AR
- I/O BR

Main Memory
- Instruction
- Instruction
- Instruction
- Data
- Data
- Data

System Bus

I/O Module
- Buffers

PC = Program counter
IR = Instruction register
MAR = Memory address register
MBR = Memory buffer register
I/O AR = Input/output address register
I/O BR = Input/output buffer register
Instruction Cycle

- Two steps:
  - Fetch
  - Execute
Fetch Cycle

- For each instruction cycle, the processor fetches an instruction from memory.
- 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

• 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.
  ▫ To control the alteration of sequence of operations.
  ▫ e.g. jump

• An instruction’s execution may involve a combination of these actions.
(a) Instruction format

(b) Integer format

Program counter (PC) = Address of instruction
Instruction register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory

(d) Partial list of opcodes

Figure 3.4 Characteristics of a Hypothetical Machine
Example of Program Execution
Instruction Cycle State Diagram

- **Instruction fetch**
- **Instruction address calculation**
- **Instruction operation decoding**
- **Operand fetch**
- **Multiple operands**
- **Operand address calculation**
- **Data Operation**
- **Operand address calculation**
- **Operand store**
- **Multiple results**

- Instruction complete, fetch next instruction
- Return for string or vector data
Instruction Cycle State

- **Instruction address calculation (iac):** Determine the address of the next instruction to be executed. Usually, this involves adding a fixed number to the address of the previous instruction.
- **Instruction fetch (if):** Read instruction from its memory location into the processor.
- **Instruction operation decoding (iod):** Analyze instruction to determine type of operation to be performed and operand(s) to be used.
- **Operand address calculation (oac):** If the operation involves reference to an operand in memory or available via I/O, then determine the address of the operand.
Instruction Cycle State

- **Operand fetch (of):** Fetch the operand from memory or read it in from I/O.
- **Data operation (do):** Perform the operation indicated in the instruction.
- **Operand store (os):** Write the result into memory or out to I/O.
Instruction Cycle State

- States in the upper part of the previous figure involve an exchange between the processor and either memory or an I/O module.
- States in the lower part of the diagram involve only internal processor operations.
- The (oac) state appears twice, because an instruction may involve a read, a write, or both.
- A single instruction can specify an operation to be performed on a vector (one-dimensional array) of numbers or a string (one-dimensional array) of characters.
Interrupts

• Mechanism by which other modules (e.g. I/O) may interrupt normal sequence of processing.

Classes of Interrupts:

• **Program**: generated by some condition that occurs as a result of an instruction execution e.g. arithmetic overflow, division by zero.

• **Timer**: generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.

• **I/O**: generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.
Interrupts

Classes of Interrupts:

- **Hardware failure**: generated by a failure such as power failure or memory parity error.

- Interrupts are provided primarily as a way to improve processing efficiency since most external devices are much slower than the processor.

- Of the next figure, The user program performs a series of WRITE calls interleaved with processing. Code segments 1, 2, and 3 refer to sequences of instructions that do not involve I/O.
Program Flow Control

(a) No interrupts
(b) Interrupts; short I/O wait
(c) Interrupts; long I/O wait
Interrupts

- The WRITE calls are to an I/O program to perform the actual I/O operation.

The I/O program consists of three sections:
- A sequence of instructions, labeled 4 in the figure, to prepare for the actual I/O operation.
- The actual I/O command.
- A sequence of instructions, labeled 5 in the figure, to complete the operation.
- Because the I/O operation may take a relatively long time to complete, the I/O program is hung up waiting for the operation to complete; hence, the user program is stopped at the point of the WRITE call for some considerable period of time.
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
• When the external device becomes ready to be, the I/O module for that external device sends an *interrupt request signal to the processor.*

• *The processor responds by* suspending operation of the current program, branching off to a program to service that particular I/O device, known as an *interrupt handler*, and resuming the original execution after the device is serviced.

• An *interrupt is just that:* an interruption of the normal sequence of execution. When the interrupt processing is completed, execution resumes.

• The processor and the operating system are responsible for suspending the user program and then resuming it at the same point.
Transfer of Control via Interrupts

User Program

Interrupt Handler

Interrupt occurs here
Instruction Cycle with Interrupts

- START
- Fetch Cycle
  - Fetch Next Instruction
- Execute Cycle
  - Execute Instruction
  - Interrupts Disabled
  - Interrupts Enabled
- Interrupt Cycle
  - Check for Interrupt; Process Interrupt
- HALT
Program Timing - Short I/O Wait

(a) Without interrupts

(b) With interrupts
Program Timing - Long I/O Wait

(a) Without interrupts

(b) With interrupts
Instruction Cycle (with Interrupts) - State Diagram
Multiple Interrupts

1. Disable interrupts (such as sequential interrupt, nested interrupt):
   ▫ Processor will ignore further interrupts while 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
Disk interrupt occurs $t=20$, because this interrupt is of lower priority, it is simply held, and the communications ISR runs to completion.
Interconnection Structures

- All the modules must be connected.
- A computer consists of a set of components or modules of three basic types (processor, memory, I/O) that communicate with each other.
- Different type of connection for different type of modules.
- The collection of paths connecting the various modules is called the *interconnection structure*. 
• **Memory:**
  - It consists of N words of equal length.
  - Each word is assigned a unique numerical address (0, 1, ..., N – 1).
  - A word of data can be read from or written into the memory.
  - The nature of the operation is indicated by read and write control signals.
  - The location for the operation is specified by an address.
• **I/O module:**
  - I/O is similar to memory.
  - There are two operations, read and write.
  - I/O module may control more than one external device.
  - We can refer to each of the interfaces to an external device as a port and give each a unique address.
  - There are external data paths for the input and output of data with an external device.
  - I/O module may be able to send interrupt signals to the processor.
• **Processor:**
  - The processor reads in instructions and data, writes out data after processing, and uses control signals to control the overall operation of the system.
  - It also receives interrupt signals.
Computer Modules

Memory

\[ \begin{align*}
N \text{ Words} \\
0 \quad \cdots \\
\cdots \\
N-1 \quad \cdots 
\end{align*} \]

I/O Module

\[ \begin{align*}
\text{Read} & \quad \text{Write} \\
\text{Address} & \quad \text{Data} \\
\text{Internal Data} & \quad \text{External Data} \\
\text{Internal Data} & \quad \text{Interrupt Signals}
\end{align*} \]

CPU

\[ \begin{align*}
\text{Instructions} & \quad \text{Address} \\
\text{Data} & \quad \text{Control Signals} \\
\text{Interrupt Signals} & \quad \text{Data}
\end{align*} \]
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.
Buses Interconnection

- There are a number of possible interconnection systems.
- Single and multiple BUS structures are most common.
- e.g. Control/Address/Data bus (PC).
- e.g. Unibus (DEC-PDP).
- A key characteristic of a bus is that it is a shared transmission medium.
- A bus consists of multiple communication pathways, or lines. Each line is capable of transmitting signals representing binary 1 and binary 0.
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.
- There are many different bus designs, on any bus the lines can be classified into three functional groups: data, address, and control lines.
Data Bus

- Carries data.
  - Remember that there is no difference between “data” and “instruction” at this level.
- Data lines provide a path for moving data among system modules. These lines are called the data bus.
- The number of lines being referred to as the width of the data bus.
- 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

- Used to control the access to and the use of the data and address lines.
- Control signals transmit both command and timing information among system modules.
- Timing signals indicate the validity of data and address information.
- Command signals specify operations to be performed.
- Control and timing information.
  - Memory read/write signal.
  - Interrupt request.
  - Clock signals.
Bus Interconnection Scheme
Operations Of Bus

• The operation of the bus is as follows:
  ▫ If one module wishes to send data to another, it must do two things:
    (1) Obtain the use of the bus.
    (2) Transfer data via the bus.

  ▫ If one module wishes to request data from another module, it must:
    (1) Obtain the use of the bus.
    (2) Transfer a request to the other module over the appropriate control and address lines. It must then wait for that second module to send the data.
Big and Yellow?

- **What do buses look like?**
  - Parallel lines on circuit boards.
  - Ribbon cables.
  - Strip connectors on mother boards.
    - e.g. PCI.
  - Sets of wires.

- The system bus is a number of parallel electrical conductors. These conductors are metal lines etched in a card or board (printed circuit board).

- In the next figure, the bus consists of two vertical columns of conductors.
Physical Realization of Bus Architecture
• Modern systems tend to have all of the major components on the same board with more elements on the same chip as the processor.

• If a component on a board fails, that board can easily be removed and replaced.
Multiple-Bus Hierarchies

Single Bus Problems

• If a great number of devices are connected to the bus, performance will suffer. There are two main causes:
  ▫ Propagation delays.
  ▫ Aggregate data.

• Most systems use multiple buses to overcome these problems.
Propagation Delay:

- The more devices attached to the bus, the greater the bus length and hence the greater the propagation delay.

- This delay determines the time it takes for devices to coordinate the use of the bus.

- When control of the bus passes from one device to another frequently, these propagation delays can noticeably affect performance.
• **Aggregate Data:**
  - The bus may become a bottleneck as the aggregate data transfer demand approaches the capacity of the bus.

• However, because the data rates generated by attached devices are growing rapidly, this is a race that a single bus is destined to lose.
Traditional bus architecture
Traditional bus architecture

- Most computer systems use multiple buses, generally laid out in a hierarchy.
- There is a local bus that connects the processor to a cache memory and that may support one or more local devices.
- The cache memory controller connects the cache not only to this local bus, but to a system bus to which are attached all of the main memory modules.
- Expansion bus interface buffers data transfers between the system bus and the I/O controllers on the expansion bus. This arrangement allows the system to support a wide variety of I/O devices.
Traditional bus architecture

• Regarding the previous Figure, it shows some typical examples of I/O devices that might be attached to the expansion bus.
• Network connections include local area networks (LANs) such as a 10-Mbps Ethernet and connections to wide area networks (WANs).
• SCSI (small computer system interface) is itself a type of bus used to support local disk drives and other peripherals.
• A serial port could be used to support a printer or scanner.
High Performance Bus Architecture
• The cache controller is integrated into a bridge, or buffering device, that connects to the high-speed bus.
• This bus supports connections to high-speed LANs, such as Fast Ethernet at 100 Mbps, video and graphics workstation controllers, as well as interface controllers to local peripheral buses, including SCSI and FireWire.
• The advantage of this arrangement is that the high-speed bus brings high demand devices into closer integration with the processor and at the same time is independent of the processor.
• Changes in processor architecture do not affect the high-speed bus, and vice versa.
### Table 3.2  Elements of Bus Design

<table>
<thead>
<tr>
<th>Type</th>
<th>Bus Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dedicated</td>
<td>Address</td>
</tr>
<tr>
<td>Multiplexed</td>
<td>Data</td>
</tr>
<tr>
<td>Method of Arbitration</td>
<td>Data Transfer Type</td>
</tr>
<tr>
<td>Centralized</td>
<td>Read</td>
</tr>
<tr>
<td>Distributed</td>
<td>Write</td>
</tr>
<tr>
<td>Timing</td>
<td>Read-modify-write</td>
</tr>
<tr>
<td>Synchronous</td>
<td>Read-after-write</td>
</tr>
<tr>
<td>Asynchronous</td>
<td>Block</td>
</tr>
</tbody>
</table>
Bus Types

- A dedicated bus line is permanently assigned either to one function or to a physical subset of computer components.

- The method of using the same lines for multiple purposes is known as time multiplexing.

- Physical dedication refers to the use of multiple buses, each of which connects only a subset of modules.
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 referred to as:
    - Bus Controller
    - Arbiter
  - Bus Controller or arbiter is responsible for allocating time on the bus.
  - The device may be a separate module or part of the processor.
    - May be part of CPU or separate.
Centralised or Distributed Arbitration

- **Distributed**
  - Each module may claim the bus.
- There is no central controller.
- Each module contains access control logic and the modules act together to share the bus.

With both methods of arbitration, the purpose is to designate one device, either the processor or an I/O module, as master. The master may initiate a data transfer (e.g., read or write) with some other device.
Timing

- **Timing** is the way in which events are coordinated on the bus.
- **Synchronous timing:**
  - Occurrence of events on the bus is determined by clock signals.
  - Control Bus includes clock line.
  - A single 1-0 transmission is a bus cycle or clock cycle.
  - All devices on the bus can read clock line.
  - All events start at the beginning of a clock cycle.
  - Usually sync on leading edge.
  - Usually a single clock cycle for an event.
Synchronous Timing Diagram

- **Clock**
- **Status lines**
- **Address lines**
- **Address enable**
- **Read cycle**
  - **Data lines**
  - **Read**
- **Write cycle**
  - **Data lines**
  - **Write**

**Timelines:**
- $T_1$
- $T_2$
- $T_3$

**Signal Phases:**
- **Status signals**
- **Stable address**
- **Valid data in**
- **Valid data out**
Asynchronous Timing

• The occurrence of one event on a bus follows and depends on the occurrence of a previous event.
• Synchronous timing is simpler to implement and test. It is less flexible than asynchronous timing. Because all devices on a synchronous bus are tied to
  • a fixed clock rate, the system cannot take advantage of advances in device performance.
• With asynchronous timing, a mixture of slow and fast devices, using older and newer technology, can share a bus.
Asynchronous Timing – Read Diagram

- Status lines
- Status signals
- Address lines
- Stable address
- Read
- Valid data
- Data lines
- Acknowledge
Asynchronous Timing – Write Diagram

- Status lines
- Address lines
- Data lines

Status signals
Stable address
Valid data
Write
Acknowledge
Bus Width

- The width of the data bus has an impact on system performance:
  - The wider the data bus, the greater the number of bits transferred at one time.

- The width of the address bus has an impact on system capacity:
  - The wider the address bus, the greater the range of locations that can be referenced.
PCI Bus

- Peripheral Component Interconnection
- Intel released to public domain
- 32 or 64 bit
- 50 lines
PCI Bus Lines (required)

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

CLK: 1 2 3 4 5 6 7 8 9

FRAME#: b h

AD: ADDRESS DATA-1 DATA-2 DATA-3

c e
c e

c e
c e
c e

C/BE#: BUS CMD Data Enable Data Enable Data Enable

d

IRDY#: Wait Data Transfer Wait Data Transfer Wait Data Transfer

TRDY#: Wait Data Transfer Wait Data Transfer Wait Data Transfer

DEVSEL#: Address Phase Data Phase Data Phase Data Phase

Wait State Wait State Wait State Wait State

Bus Transaction
PCI Bus Arbiter
PCI Bus Arbitration

CLK
REQ#-A
REQ#-B
GNT#-A
GNT#-B
FRAME#
IRDY#
TRDY#
AD

Address
Data
access-A
Address
Data
access-B
Foreground Reading

- Stallings, chapter 3 (all of it)
- www.pcguide.com/ref/mbsys/buses/

- In fact, read the whole site!
- www.pcguide.com/