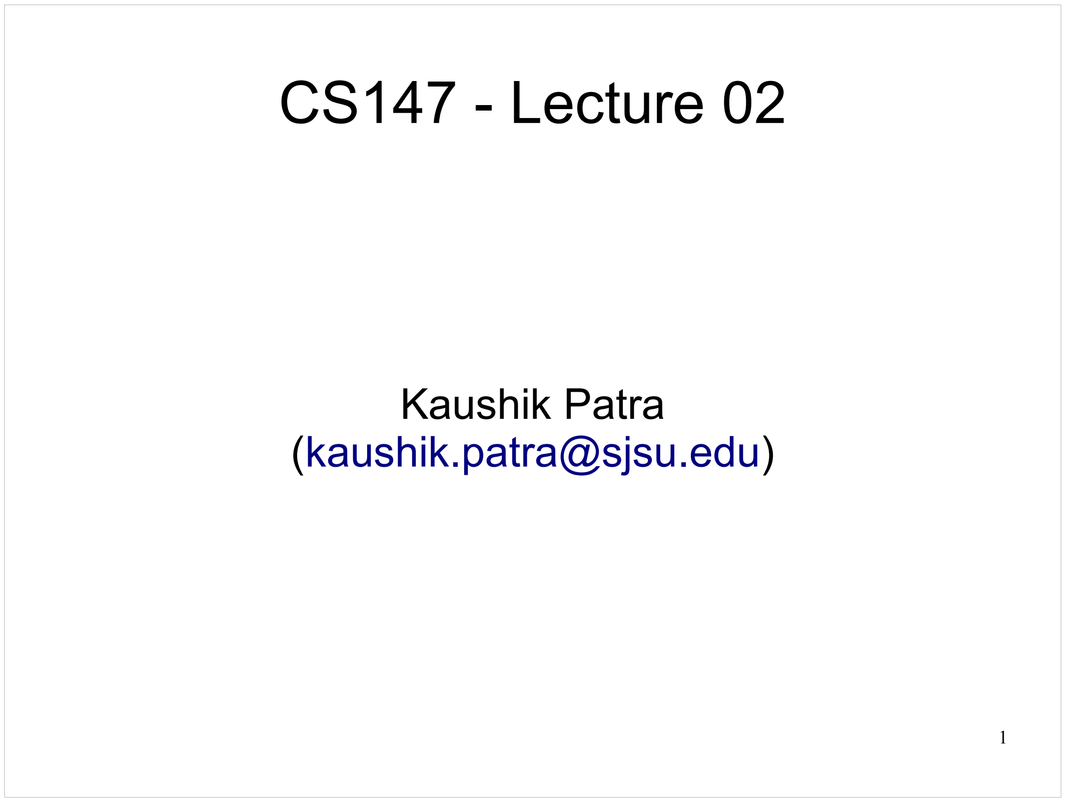


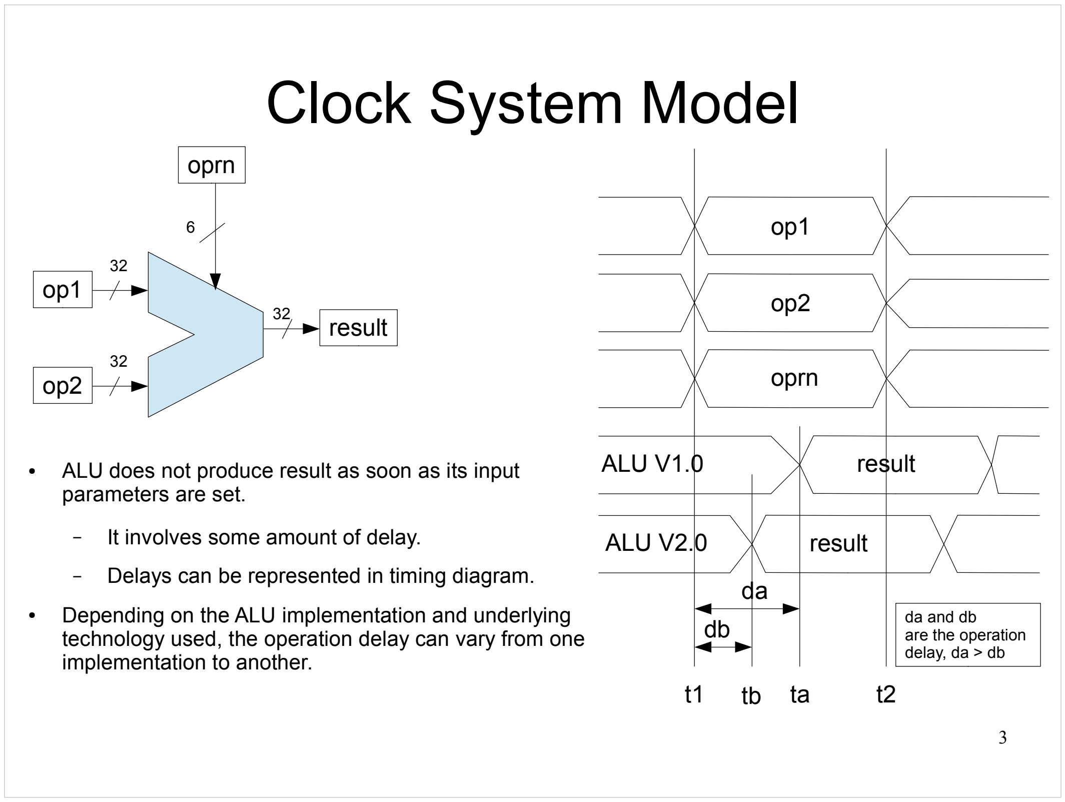
CS147 - Lecture 02

Kaushik Patra
(kaushik.patra@sjsu.edu)

1

- Topics
  - Clock System Model
  - Memory System Model
  - Control Unit Model
  - Von Neumann Architecture
  - System Software

Clock System Model …



Clock System Model

oprn
6
32
op1
32
result
32
op2
op1
op2
oprn
ALU V1.0
result
ALU V2.0
result
da
db
da and db are the operation delay, da > db
t1
tb
ta
t2

- ALU does not produce result as soon as its input parameters are set.
  - It involves some amount of delay.
  - Delays can be represented in timing diagram.
- Depending on the ALU implementation and underlying technology used, the operation delay can vary from one implementation to another.

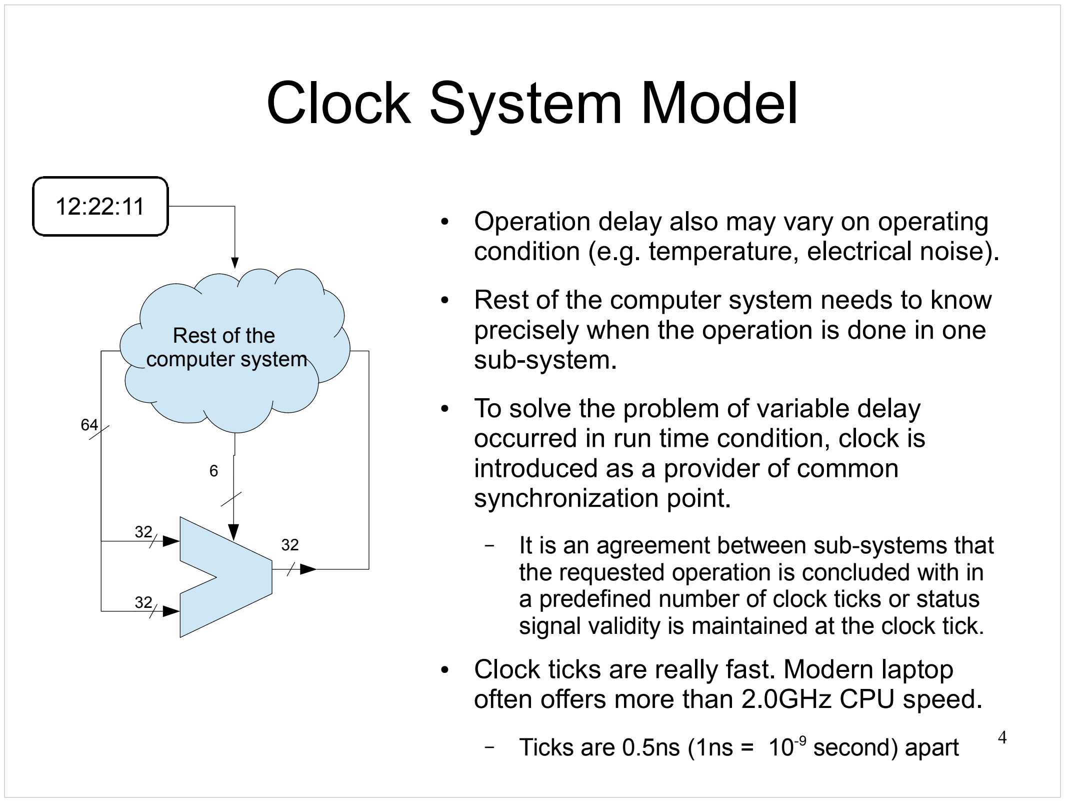
3

- Speed of operation for any digital logic circuit is limited by the underlying devices and technology that implements it. It is a part of electrical engineering study to learn techniques to speed up logic operations.

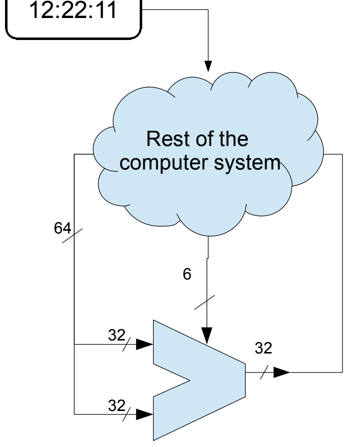
- Speed of operation also depends on the implementation of the logic circuit. Like a software implementation, one set of hardware objective (e.g. providing ALU functionality) can be done through different implementation. It is part of computer science study to improve algorithms for faster operation.

- Timing diagram depicts change of values in parameters or signals in association with time. Simplistic model assumes that the value change is instantaneous. However, in reality there are delay associated even with value change.

- Group of multiple signal (or bus) is depicted with hexagon in timing diagram, where single signal is depicted with single line.



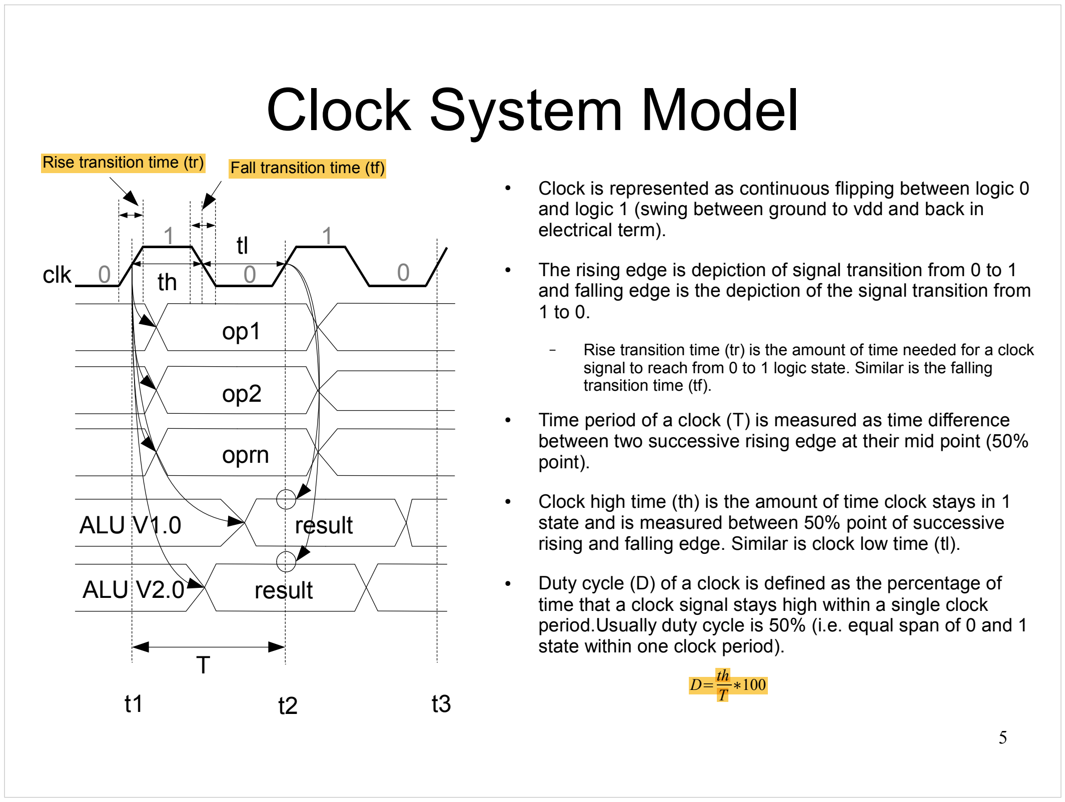
Clock System Model



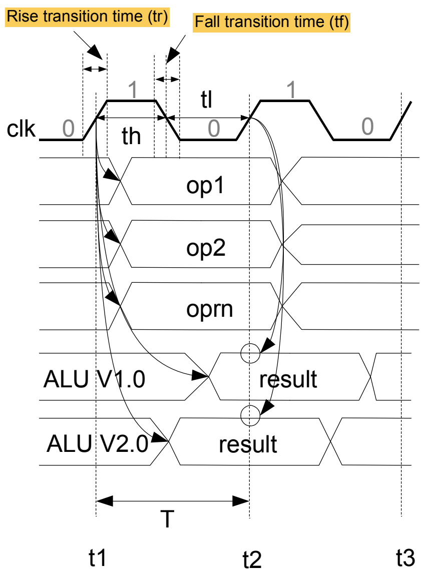
- Operation delay also may vary on operating condition (e.g. temperature, electrical noise).
- Rest of the computer system needs to know precisely when the operation is done in one sub-system.
- To solve the problem of variable delay occurred in run time condition, clock is introduced as a provider of common synchronization point.
  - It is an agreement between sub-systems that the requested operation is concluded with in a predefined number of clock ticks or status signal validity is maintained at the clock tick.
- Clock ticks are really fast. Modern laptop often offers more than 2.0GHz CPU speed.
  - Ticks are 0.5ns (1ns = 10^{-9} second) apart

4

- All the subsystems of a computer synchronizes their operation at the clock tick. In digital logic term, this type of circuits involving synchronization by clock is known as 'synchronous' circuit.
- There is another class of digital design not involving common synchronization point, known as 'asynchronous circuit'.
- Though asynchronous circuit can, theoretically, achieve much higher speed of operation, it is very hard to implement complex yet highly reliable circuit with that technique. On the other hand, though we are sacrificing operation performance, it is easier to implement synchronous circuit with predictable behavior.
- Distance between the clock ticks (or time period) of clock is defined as reciprocal of clock frequency (**T = 1/f**).



Clock System Model



- Clock is represented as continuous flipping between logic 0 and logic 1 (swing between ground to vdd and back in electrical term).
- The rising edge is depiction of signal transition from 0 to 1 and falling edge is the depiction of the signal transition from 1 to 0.
  - Rise transition time (tr) is the amount of time needed for a clock signal to reach from 0 to 1 logic state. Similar is the falling transition time (tf).
- Time period of a clock (T) is measured as time difference between two successive rising edge at their mid point (50% point).
- Clock high time (th) is the amount of time clock stays in 1 state and is measured between 50% point of successive rising and falling edge. Similar is clock low time (tl).
- Duty cycle (D) of a clock is defined as the percentage of time that a clock signal stays high within a single clock period.Usually duty cycle is 50% (i.e. equal span of 0 and 1 state within one clock period).

$$D=\frac{th}{T}*100$$

5

- Clock in real circuit is made using quartz crystal. Its mechanical resonance of vibration is translated into electrical signal oscillation through electronic oscillation circuit. This mechanism can produce precise amount of clock frequency needed for the target system.

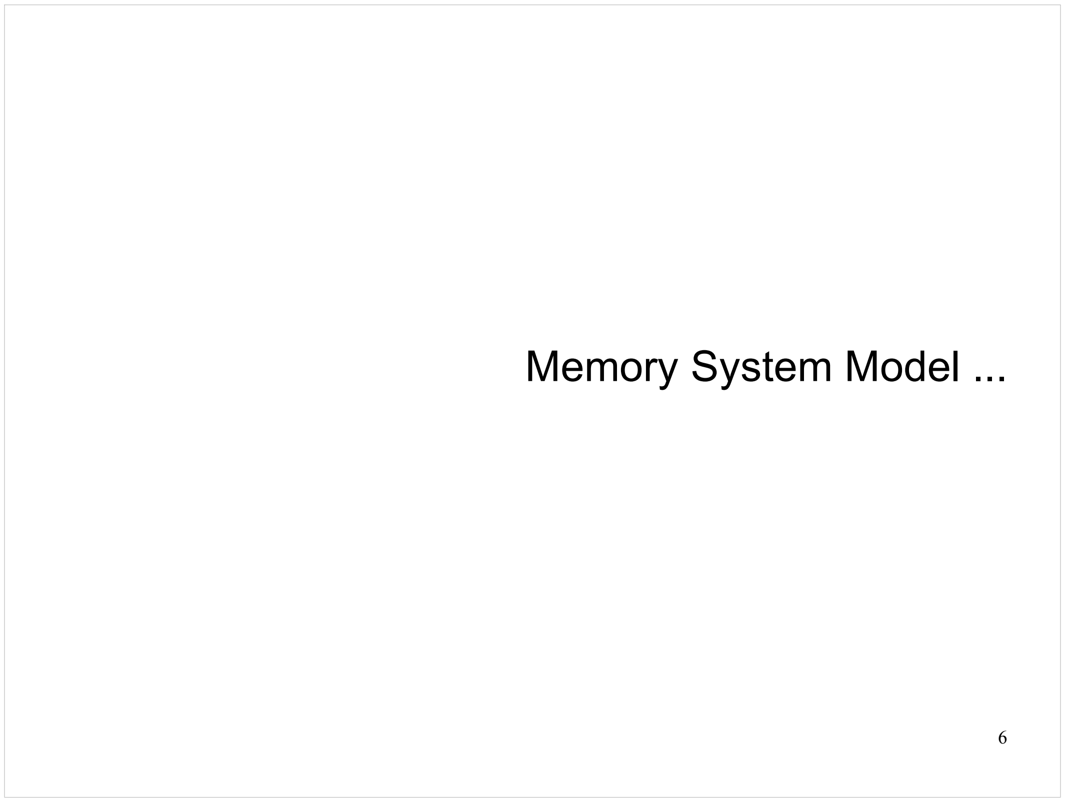
- Time period T is sum of clock high time and low time (T = th + tl).

- In the timing diagram it is shown using connecting arrows that rising clock edge at t1 triggered the change in op1, op2, oprn, and result. At the rising edge at t2 the result is sampled / used.

  - We usually use this type of relational arrow for multi clock cycle operations to clearly denote which clock edge triggered what event.

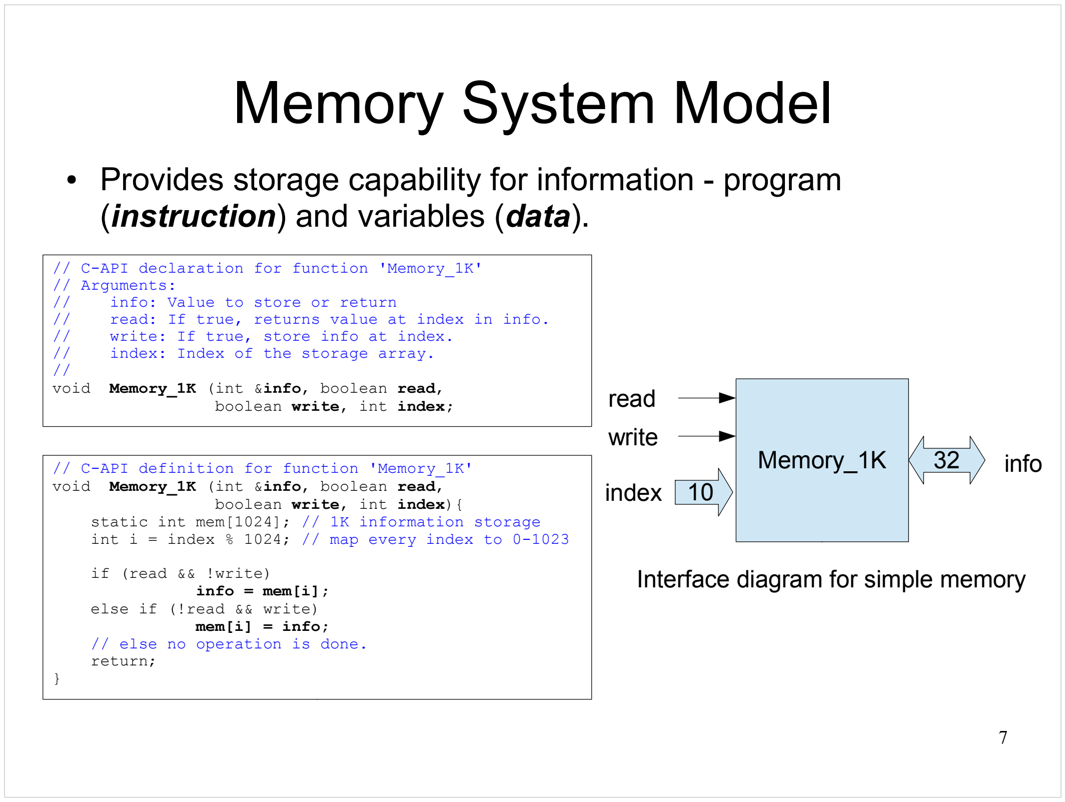
- Operation can be synchronized at either rising or falling edge.

- Since result is sampled at certain predefined interval, synchronous circuit sacrifices certain amount of performance (since it is not using the result as soon as it is ready). However, synchronous design can give much more stability to the digital circuit.



Memory System Model ...

6



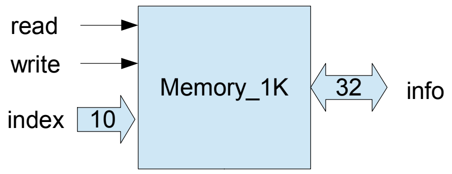
Memory System Model

- Provides storage capability for information - program (***instruction***) and variables (***data***).

```
// C-API declaration for function 'Memory_1K'
// Arguments:
//    info: Value to store or return
//    read: If true, returns value at index in info.
//    write: If true, store info at index.
//    index: Index of the storage array.
//
void  Memory_1K (int &info, boolean read,
                 boolean write, int index;
```

```
// C-API definition for function 'Memory_1K'
void  Memory_1K (int &info, boolean read,
                 boolean write, int index){
    static int mem[1024]; // 1K information storage
    int i = index % 1024; // map every index to 0-1023

    if (read && !write)
               info = mem[i];
    else if (!read && write)
               mem[i] = info;
    // else no operation is done.
    return;
}
```



Interface diagram for simple memory

7

- static variable within a function defines a non-volatile storage within scope of that function.

- There is no distinction between 'instruction' and 'data' in context of storage. Any one of them is 'data' from memory perspective. They are just a sequence of bit values. We usually call 'info' as 'data' in general in memory context.

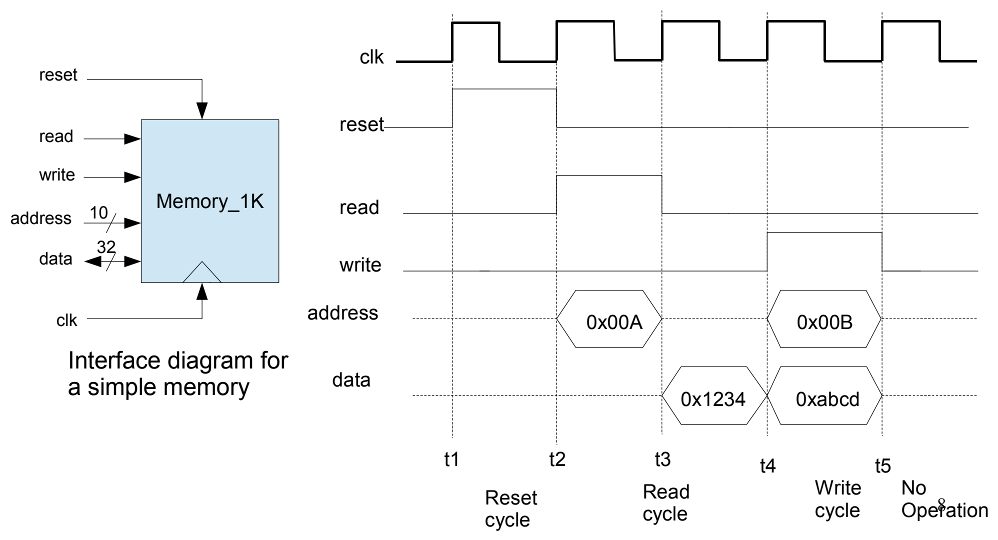
- In context of memory, the 'index' is usually called 'address'.

- There is usually additional 'reset' pin to clear memory content and set all index position to 0.

- This is very simplistic model of memory where transaction happens as 32-bit long word. In most cases, memories are byte addressable (i.e. specific byte in an word can be read/write).

Memory System Model

- Let's add reset and clock pin and review timing diagram.

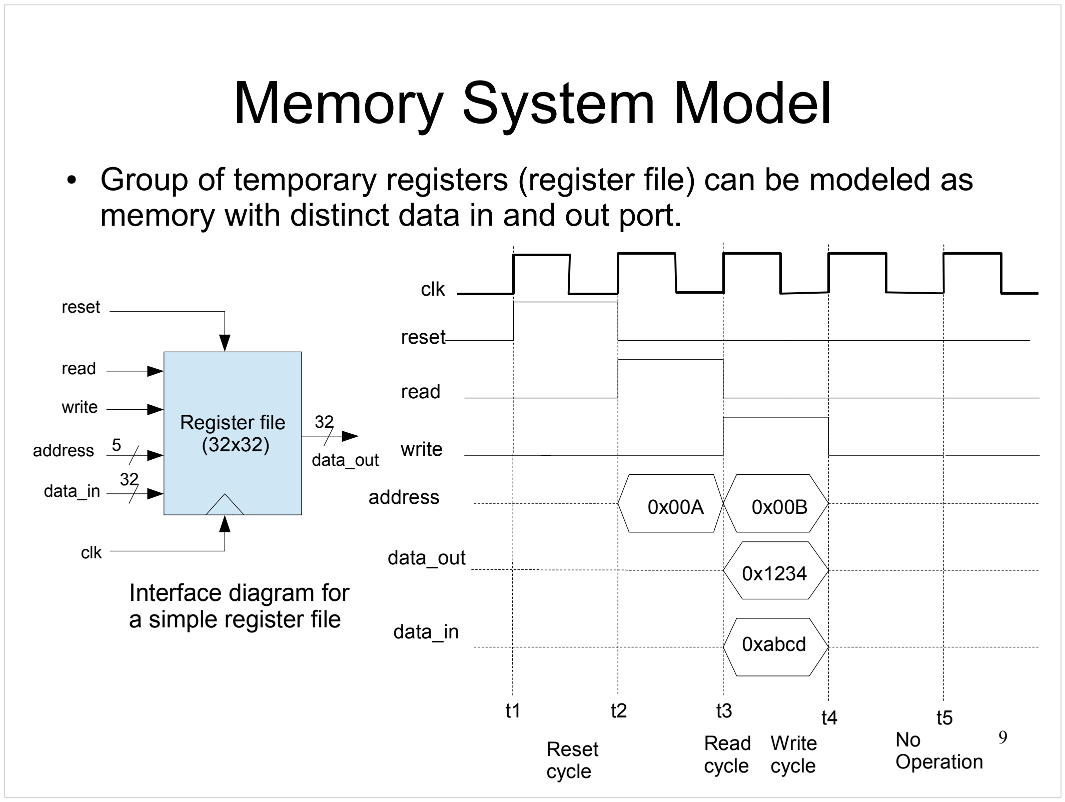


• When both read and write are of same value, the data bus is on High-Z state which means it is electrically detached (open circuit) from rest of the circuit.

• The timing diagram shows that at t1 time reset operation is done following by a read operation at t2 and write operation at t4.

• This memory provides the instruction and data storage for a program to run on a processor.

• Memory may need multiple clock cycle to complete its operation. We have modeled a single clock cycle access to simplify system implementation.



Memory System Model

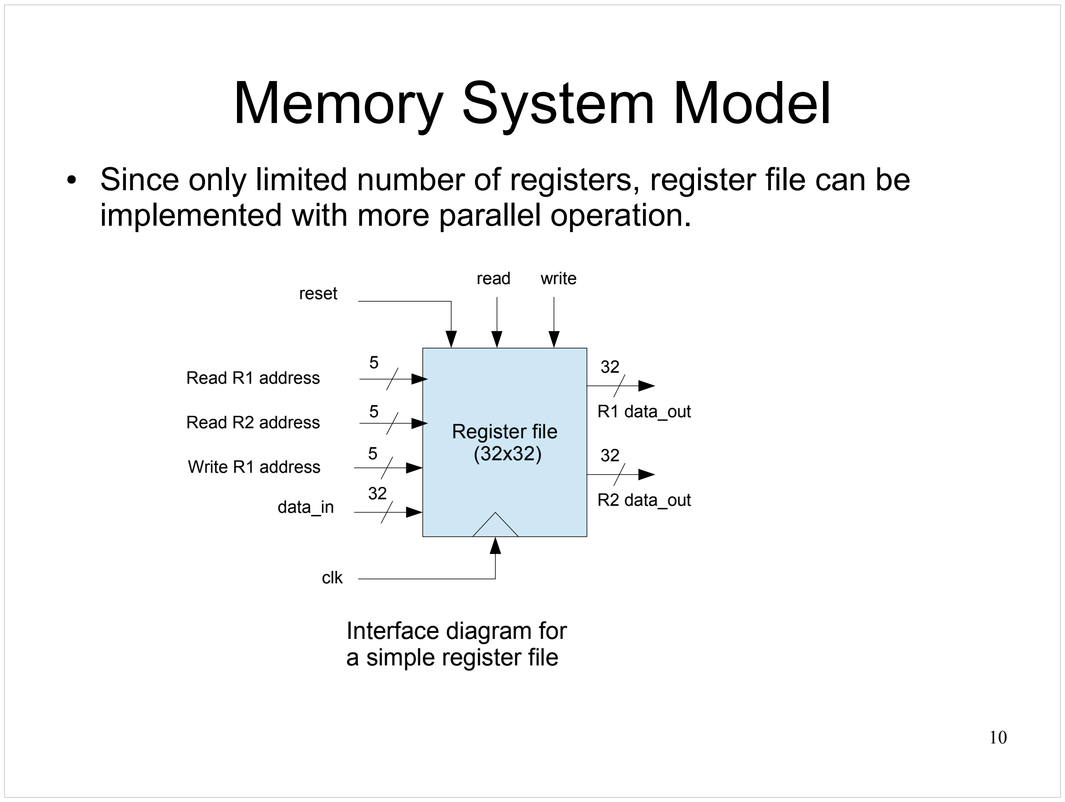
- Group of temporary registers (register file) can be modeled as memory with distinct data in and out port.

- When not reading, the data_out remains High-Z.

- Register file is given separate data in and data out to simplify implementation of processor, unlike the standard memory model. Standard memory model targets to save some space and wiring using common in-out data bus for data operation.

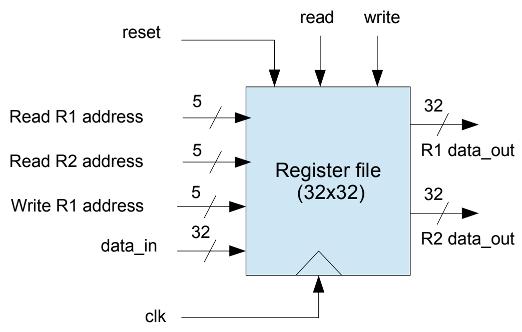
- Register file is usually a part of processor, where standard memory is modeled outside processor.

  - We'll review concept of 'cache' memory which sits inside processor as an integrated part of it.



Memory System Model

- Since only limited number of registers, register file can be implemented with more parallel operation.



Interface diagram for a simple register file

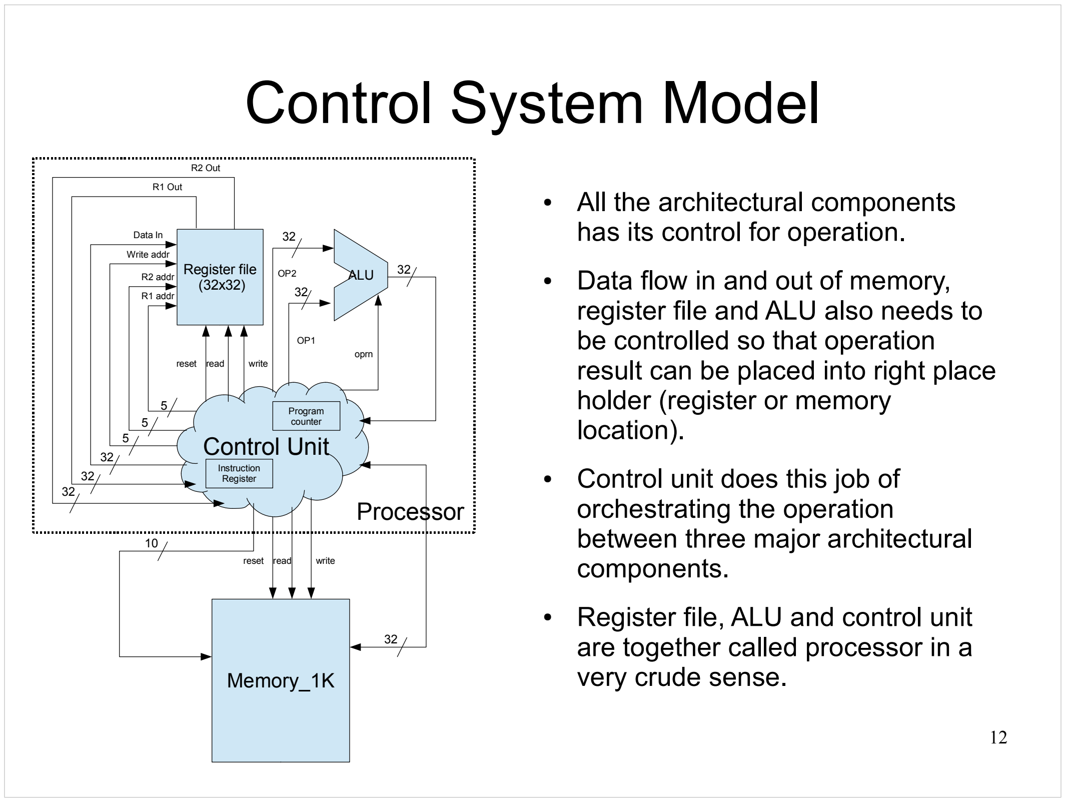
10

- Since our instruction set 'cs147sec05' has fixed position for the register operands, it will be easier and faster to access both the register at once.

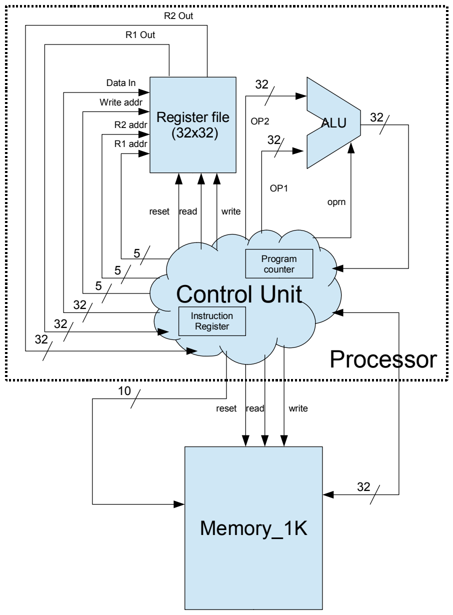
  - This is instruction set driven architecture.

  - If the instruction set does not provide such uniformity, there would no such simplification for faster operation.

Control System Model ...



Control System Model



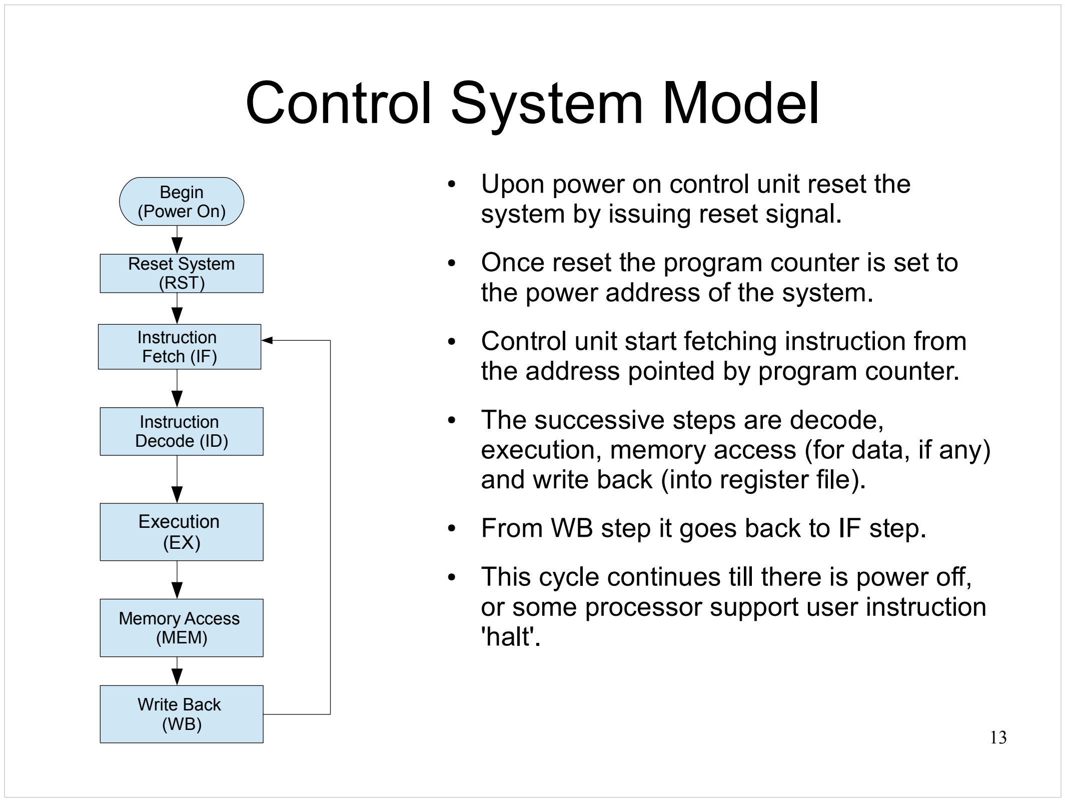
- All the architectural components has its control for operation.
- Data flow in and out of memory, register file and ALU also needs to be controlled so that operation result can be placed into right place holder (register or memory location).
- Control unit does this job of orchestrating the operation between three major architectural components.
- Register file, ALU and control unit are together called processor in a very crude sense.

12

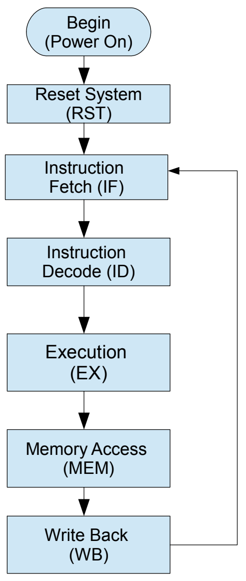
- Both register file and memory has reset, read and write signal. ALU has 'oprn' signal to select the operation.

- Data flow occurs between
  - Register file to ALU as op1 and op2
  - ALU to register file as result.
  - Register file to memory to store result.
  - Memory to register file to load value from memory.

- Control unit contains two special registers.
  - Instruction register to hold current instruction after fetching from memory.
  - Program counter to hold memory address to fetch next instruction.



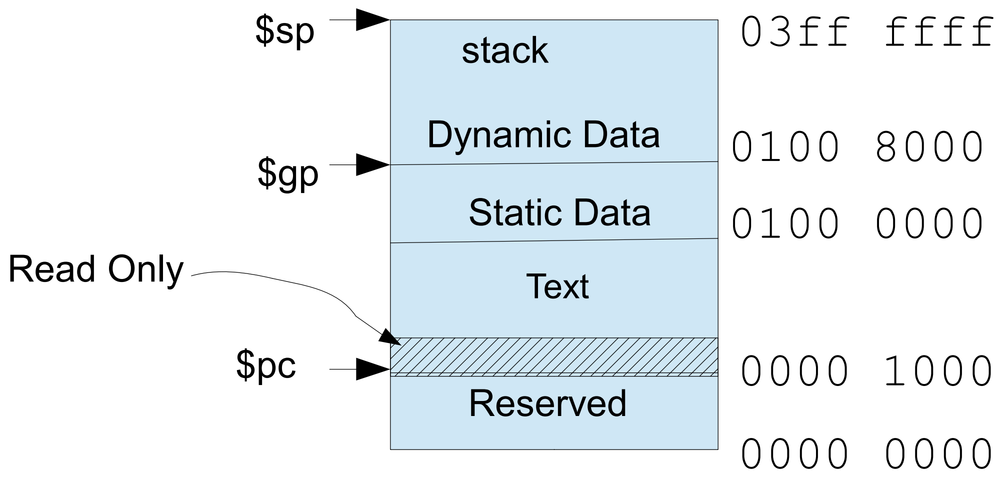
Control System Model



- Upon power on control unit reset the system by issuing reset signal.
- Once reset the program counter is set to the power address of the system.
- Control unit start fetching instruction from the address pointed by program counter.
- The successive steps are decode, execution, memory access (for data, if any) and write back (into register file).
- From WB step it goes back to IF step.
- This cycle continues till there is power off, or some processor support user instruction 'halt'.

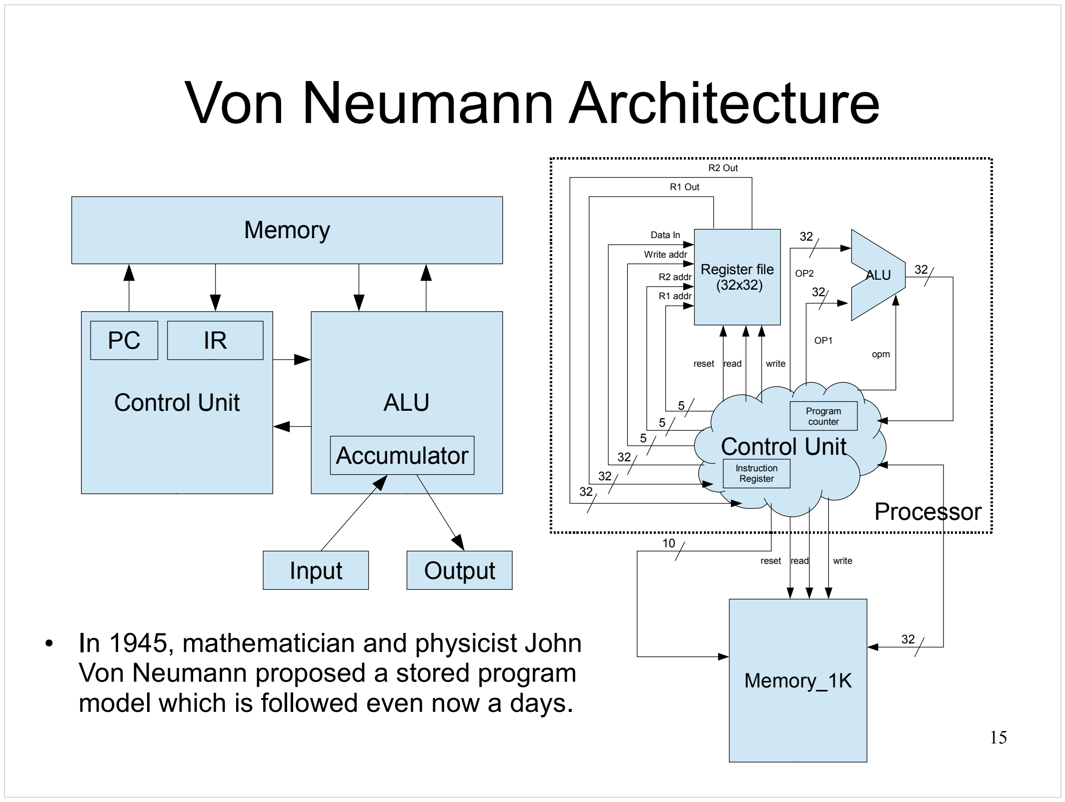
13

- Power on address is predefined for a system.
- A part of memory starting from the power address are implemented as read-only memory (ROM) which can not be rest with a reset signal. This read-only memory part contains initial set of instructions to load BOOT program (from disk or other semi permanent memory) at the start of volatile part of the system memory (RAM) within the text area of the system memory map.



Von Neumann Architecture ...

14

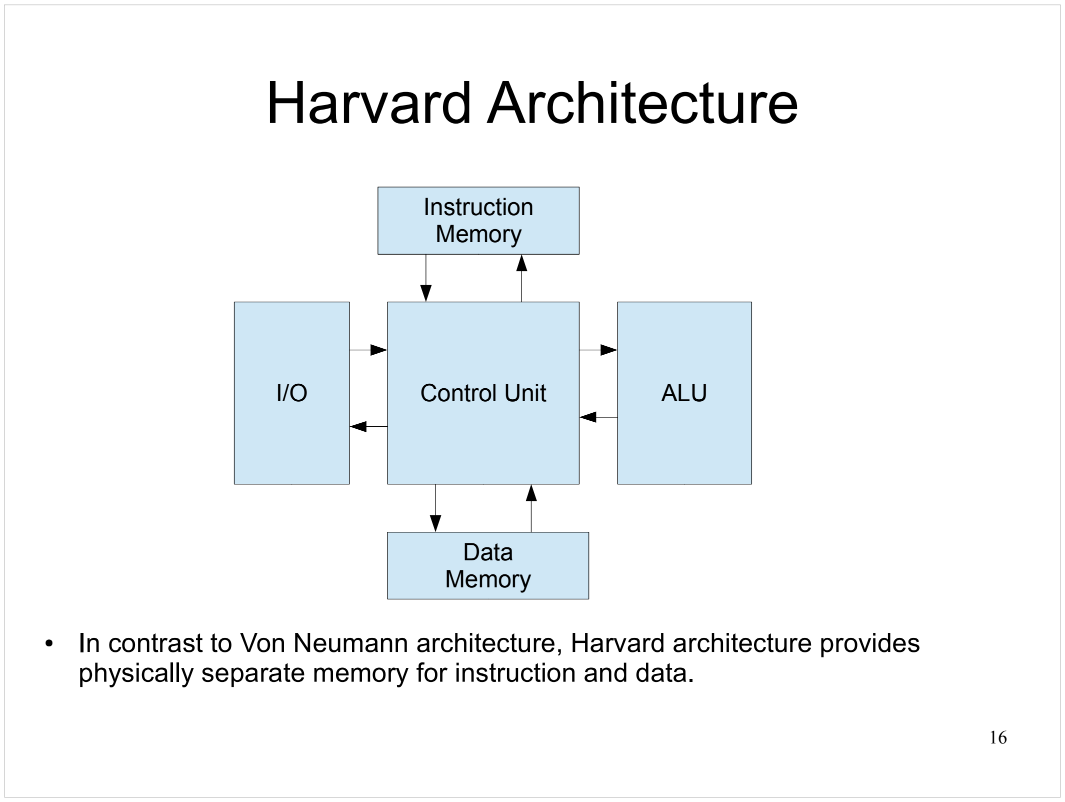


Von Neumann Architecture

- In 1945, mathematician and physicist John Von Neumann proposed a stored program model which is followed even now a days.

15

- Stored program concept was first proposed by this architecture.
- Control unit has program counter register and instruction register just like any modern day processor.
- ALU has one special register 'accumulator' which stores any input from user. This register also stores result of any ALU operation to send into any output device.
- In this architecture instruction and data for operation must be fetched in different cycle since they both uses same memory.



Harvard Architecture

Instruction Memory
I/O
Control Unit
ALU
Data Memory

- In contrast to Von Neumann architecture, Harvard architecture provides physically separate memory for instruction and data.

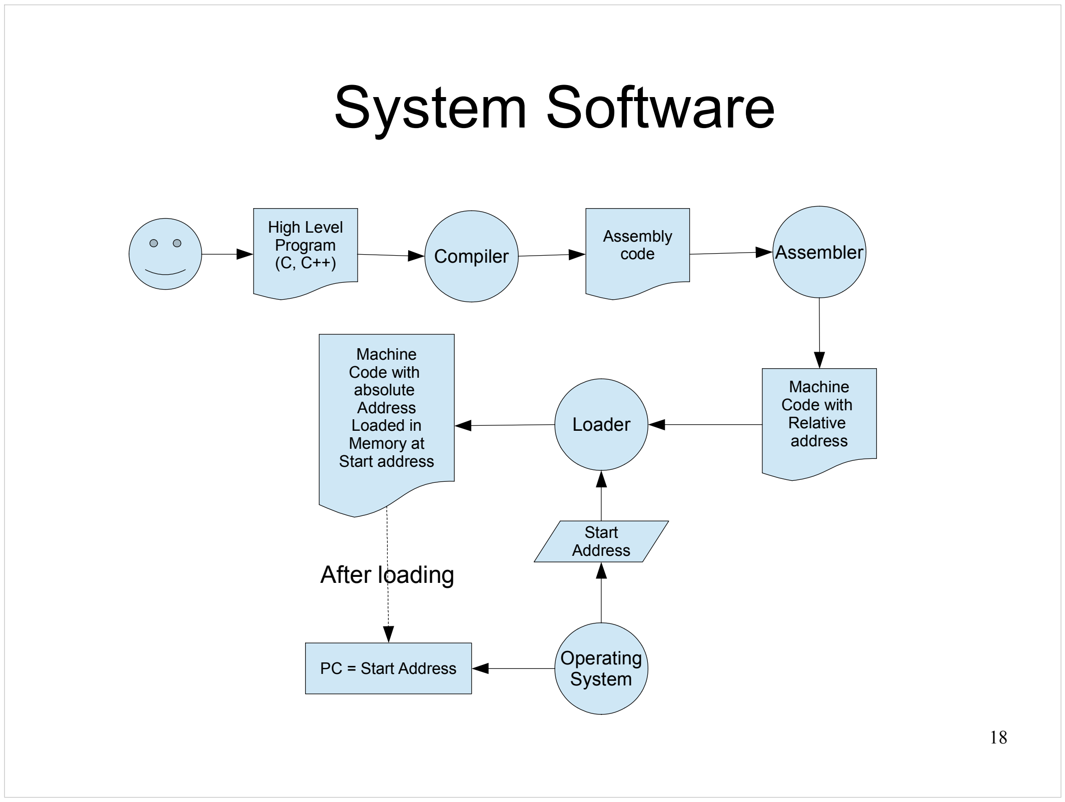
16

- Harvard architecture provides distinct address space for memory and data. Modified Harvard architecture allows to have single physical memory but the instruction and data access is still in parallel to each other if needed.

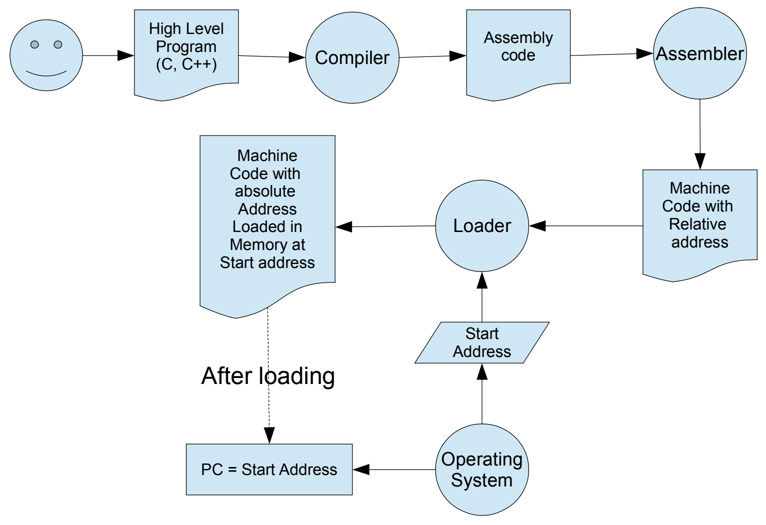
- Many modern application for Harvard architecture including processor cache mechanism with separate cache for instruction and data. Other application area includes DSP (Digital Signal Processing) and micro-controller.

System Software ...

17



System Software


High Level Program (C, C++)
Compiler
Assembly code
Assembler
Machine Code with Relative address
Loader
Machine Code with absolute Address Loaded in Memory at Start address
Start Address
After loading
PC = Start Address
Operating System

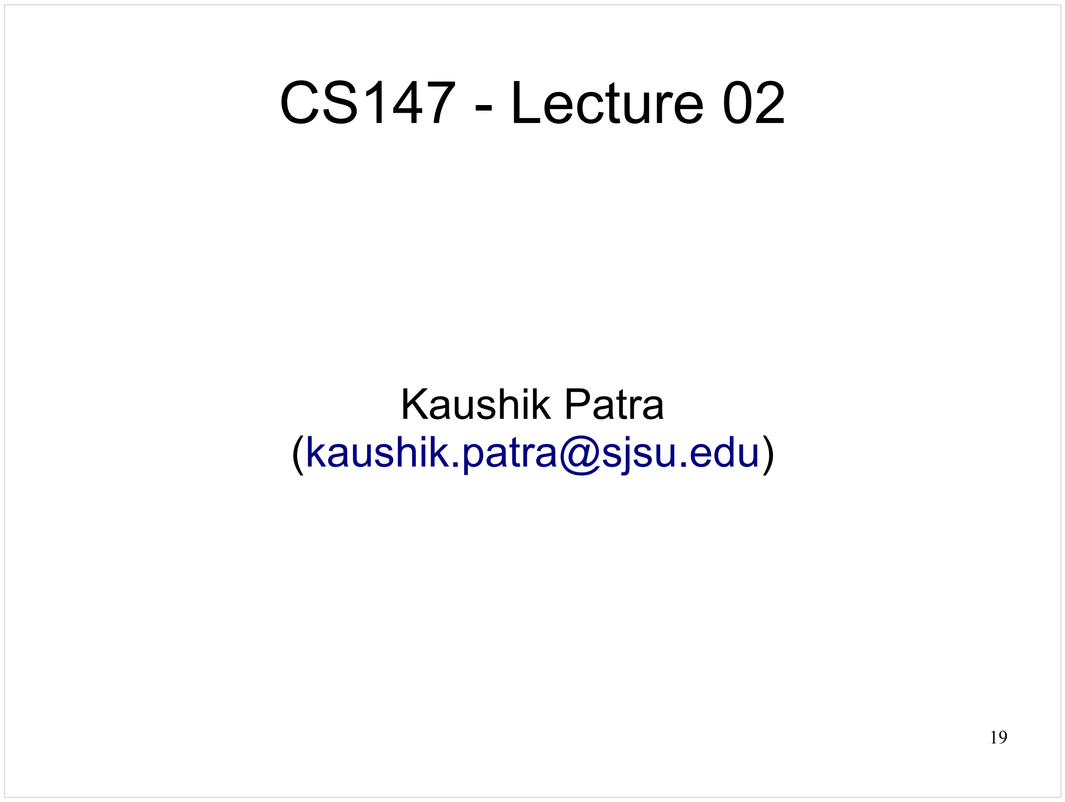
18

• Compiler process a program in high level language and generates equivalent program using assembly language of the target system.

• The assembler translate the assembly program into machine codes with relative address. The address where the program will be loaded is not known at assembling time, hence it needs to use relative address.

• The compiling and assembling is usually done by single compilation command. The assembly code is intermediate and often reside in memory while compiler is running. Optionally we can dump assembly program from compiler. The output of a compiler command is usually the machine code with relative address.

• Once user requests OS to execute a compiled / assembled program, OS creates new process information including process id, address space, start address for loading. It calls loader with the start address. The loader program reads in the assembled user program and translates relative address into absolute address before loading it into memory at the start address specified by OS. Once loaded, OS sets the program counter (PC) to point to the start address of the user program. Once PC is set to point to user program code, the user program gets executed.



CS147 - Lecture 02

Kaushik Patra
(kaushik.patra@sjsu.edu)

19

- Topics
  - Clock System Model
  - Memory System Model
  - Control Unit Model
  - Von Neumann Architecture
  - System Software