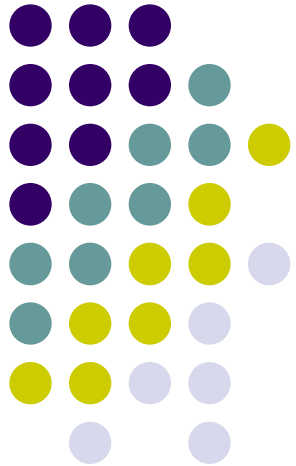
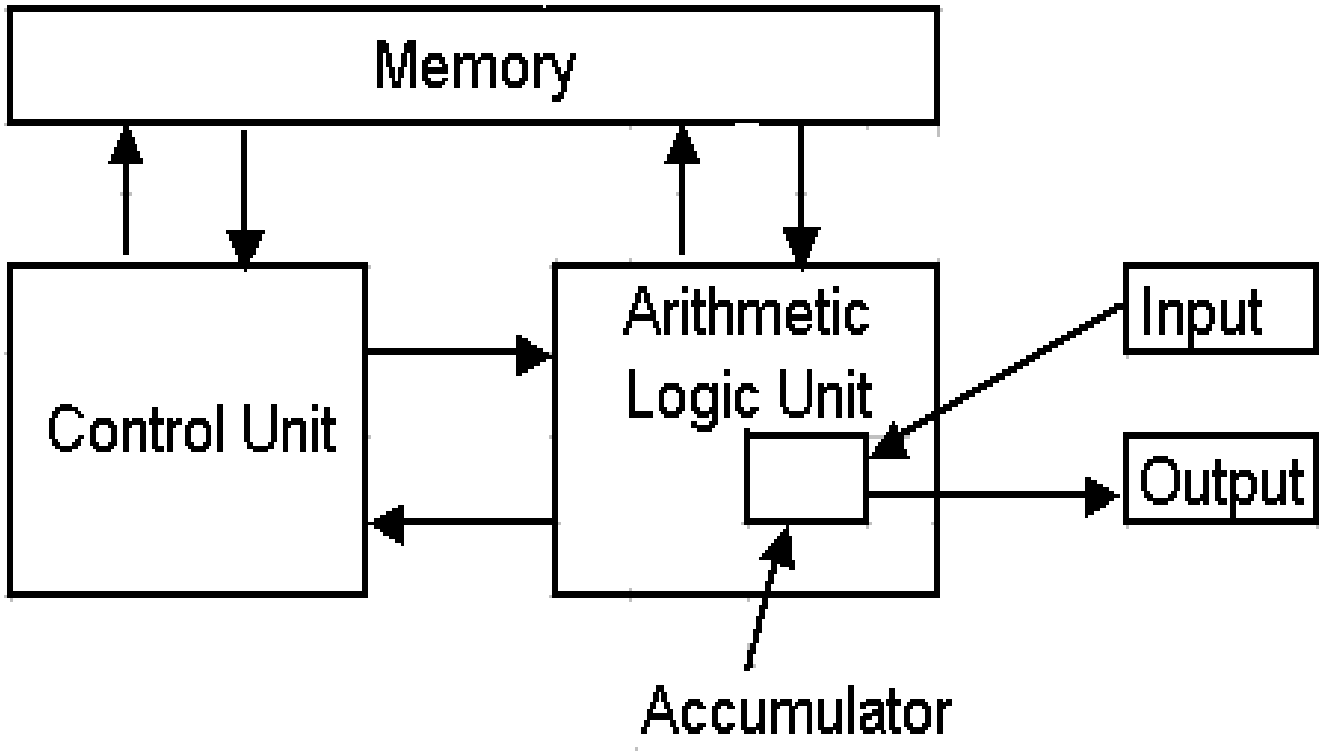
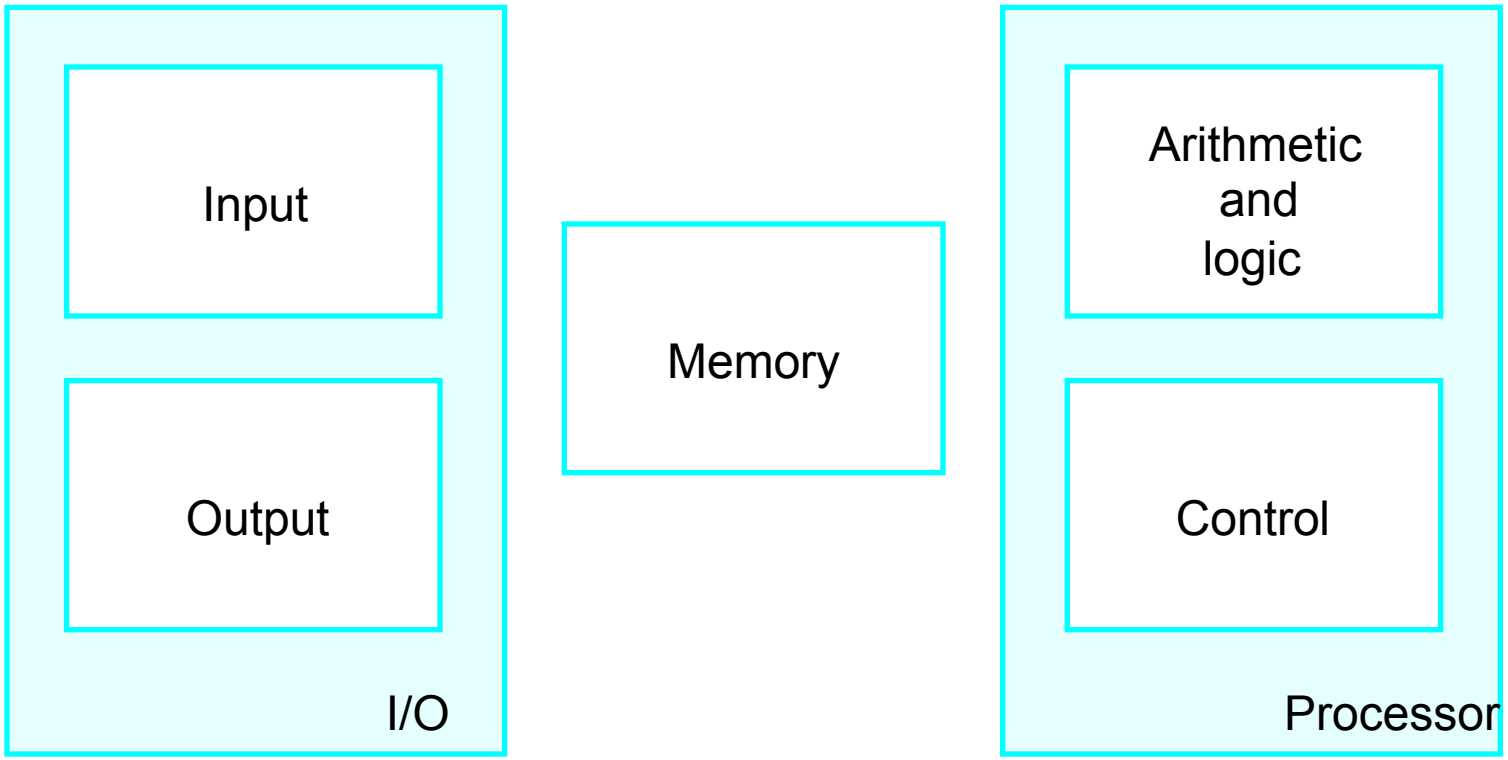
Functional Units



The Original Von Neumann Machine

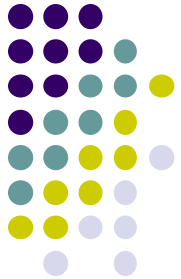


Functional Units

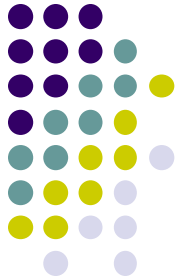


Basic functional units of a computer.

Information Handled by a Computer



- Instructions/machine instructions
  - Govern the transfer of information within a computer as well as between the computer and its I/O devices
  - Specify the arithmetic and logic operations to be performed
  - Program
- Data
  - Used as operands by the instructions
  - Source program
- Encoded in binary code – 0 and 1



- Binary Coded Decimal (Each decimal digit encoded in 4 bit)
- American Standard Code for Information Interchange (Each character is encoded in 7 bits)
- Extended Binary-Coded Decimal Interchange(8 bit are used to denote a character)

BCD or 8421 code:-

It is composed of four bits representing the decimal digits 0 through 9. The 8421 indicates the binary weights of the four bits(2^3,2^2,2^1,2^0).

Decimal	8421(BCD)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

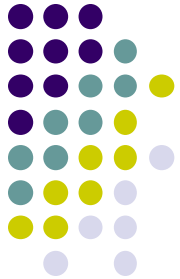
American Standard Code for Information Interchange

Bit positions	Bit positions 654							
3210	000	001	010	011	100	101	110	111
0000	NUL	DLE	SPACE	0	@	P	´	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	”	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	’	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	/	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	—	o	DEL

NUL	Null/Idle	SI	Shift in
SOH	Start of header	DLE	Data link escape
STX	Start of text	DC1-DC4	Device control
ETX	End of text	NAK	Negative acknowledgment
EOT	End of transmission	SYN	Synchronous idle
ENQ	Enquiry	ETB	End of transmitted block
ACK	Acknowledgment	CAN	Cancel (error in data)
BEL	Audible signal	EM	End of medium
BS	Back space	SUB	Special sequence
HT	Horizontal tab	ESC	Escape
LF	Line feed	FS	File separator
VT	Vertical tab	GS	Group separator
FF	Form feed	RS	Record separator
CR	Carriage return	US	Unit separator
SO	Shift out	DEL	Delete/Idle

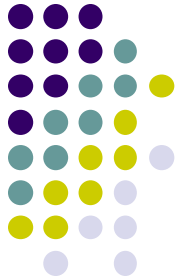
Bit positions of code format = | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Memory Unit



- Store programs and data
- Two classes of storage
  - Primary storage
    - Fast
    - Programs must be stored in memory while they are being executed
    - Large number of semiconductor storage cells
    - Processed in words
    - Address
    - RAM and memory access time
    - Memory hierarchy – cache, main memory
  - Secondary storage – larger and cheaper

Arithmetic and Logic Unit (ALU)

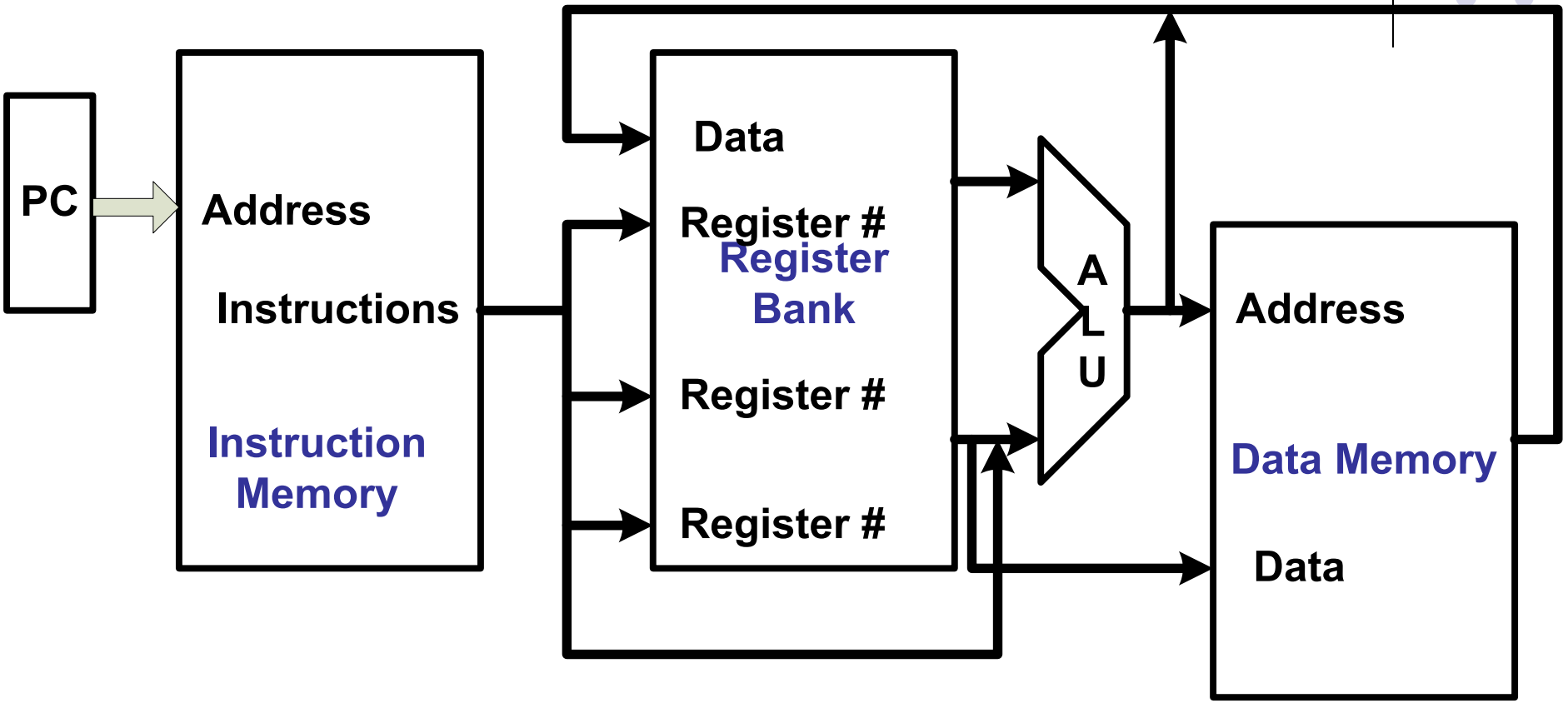


- Most computer operations are executed in ALU of the processor.
- Load the operands into memory – bring them to the processor – perform operation in ALU – store the result back to memory or retain in the processor.
- Registers
- Fast control of ALU

Control Unit

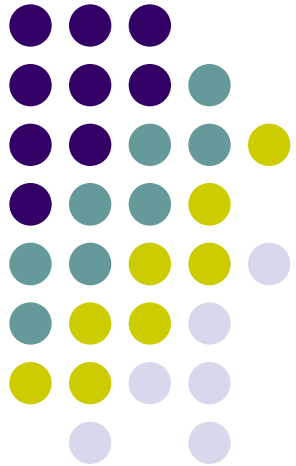
- All computer operations are controlled by the control unit.
- The timing signals that govern the I/O transfers are also generated by the control unit.
- Control unit is usually distributed throughout the machine instead of standing alone.
- Operations of a computer:
  - Accept information in the form of programs and data through an input unit and store it in the memory
  - Fetch the information stored in the memory, under program control, into an ALU, where the information is processed
  - Output the processed information through an output unit
  - Control all activities inside the machine through a control unit

The processor : Data Path and Control

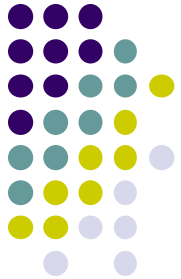


- Two types of functional units:
  - elements that operate on data values (combinational)
  - elements that contain state (state elements)

Basic Operational Concepts

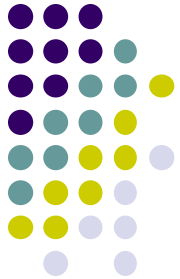


Review



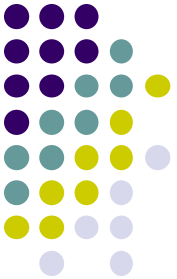
- Activity in a computer is governed by instructions.
- To perform a task, an appropriate program consisting of a list of instructions is stored in the memory.
- Individual instructions are brought from the memory into the processor, which executes the specified operations.
- Data to be used as operands are also stored in the memory.

A Typical Instruction



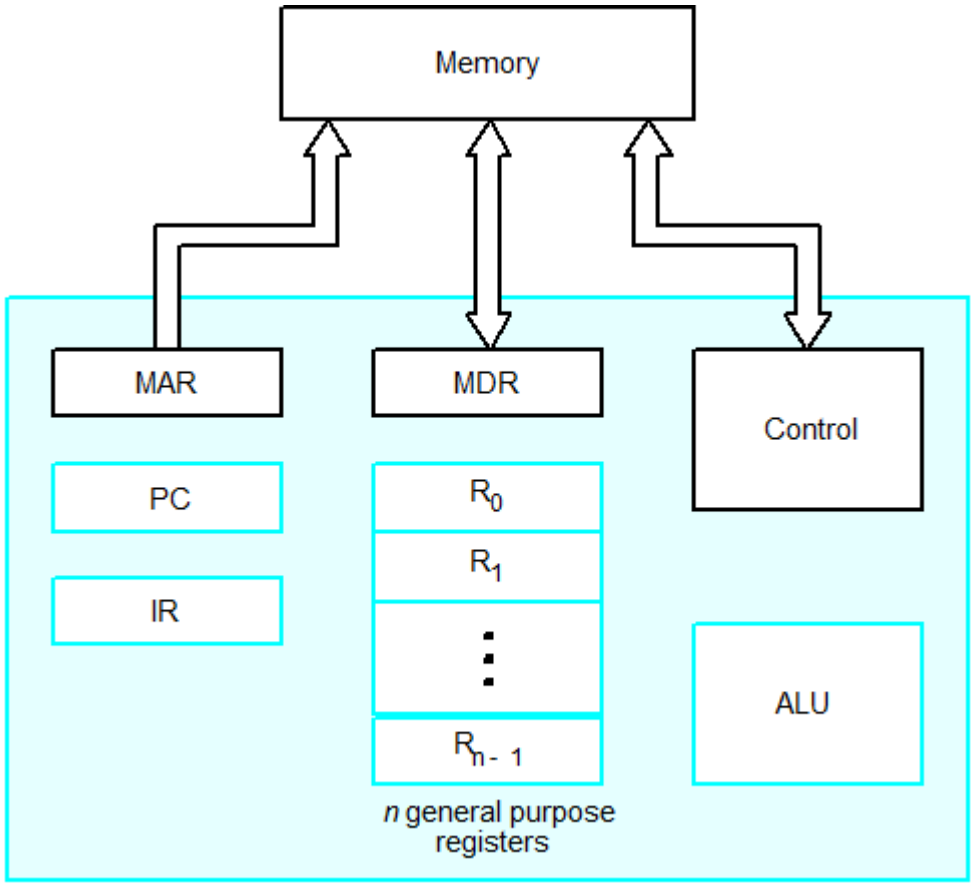
- Add LOCA, R0
- Add the operand at memory location LOCA to the operand in a register R0 in the processor.
- Place the sum into register R0.
- The original contents of LOCA are preserved.
- The original contents of R0 is overwritten.
- Instruction is fetched from the memory into the processor – the operand at LOCA is fetched and added to the contents of R0 – the resulting sum is stored in register R0.

Separate Memory Access and ALU Operation

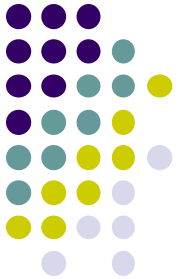


- Load LOCA, R1
- Add R1, R0
- Whose contents will be overwritten?

Connection Between the Processor and the Memory

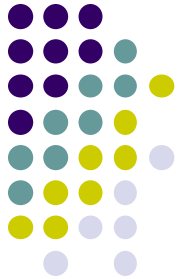


Registers



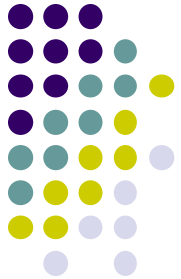
- Instruction register (IR)
- Program counter (PC)
- General-purpose register (R_0 – R_{n-1})
- Memory address register (MAR)
- Memory data register (MDR)

Typical Operating Steps



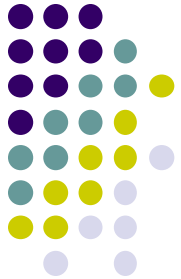
- Programs reside in the memory through input devices
- PC is set to point to the first instruction
- The contents of PC are transferred to MAR
- A Read signal is sent to the memory
- The first instruction is read out and loaded into MDR
- The contents of MDR are transferred to IR
- Decode and execute the instruction

Typical Operating Steps (Cont')



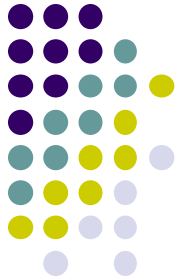
- Get operands for ALU
  - General-purpose register
  - Memory (address to MAR – Read – MDR to ALU)
- Perform operation in ALU
- Store the result back
  - To general-purpose register
  - To memory (address to MAR, result to MDR – Write)
- During the execution, PC is incremented to the next instruction

Interrupt



- Normal execution of programs may be preempted if some device requires urgent servicing.
- The normal execution of the current program must be interrupted – the device raises an *interrupt* signal.
- Interrupt-service routine
- Current system information backup and restore (PC, general-purpose registers, control information, specific information)

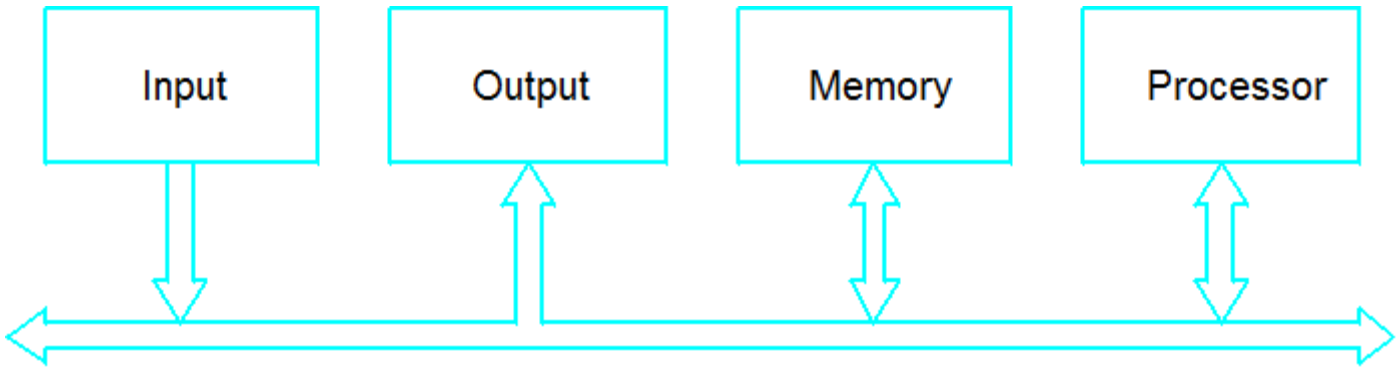
Bus Structures



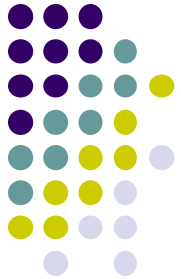
- There are many ways to connect different parts inside a computer together.
- A group of lines that serves as a connecting path for several devices is called a *bus*.
- Address/data/control

Bus Structure

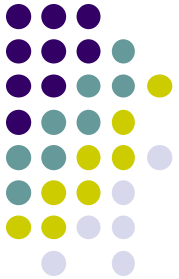
- Single-bus



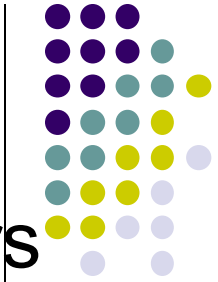
Speed Issue



- Different devices have different transfer/operate speed.
- If the speed of bus is bounded by the slowest device connected to it, the efficiency will be very low.
- How to solve this?
- A common approach – use buffers.

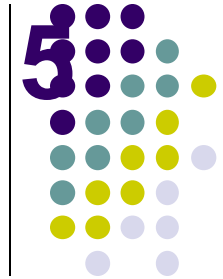


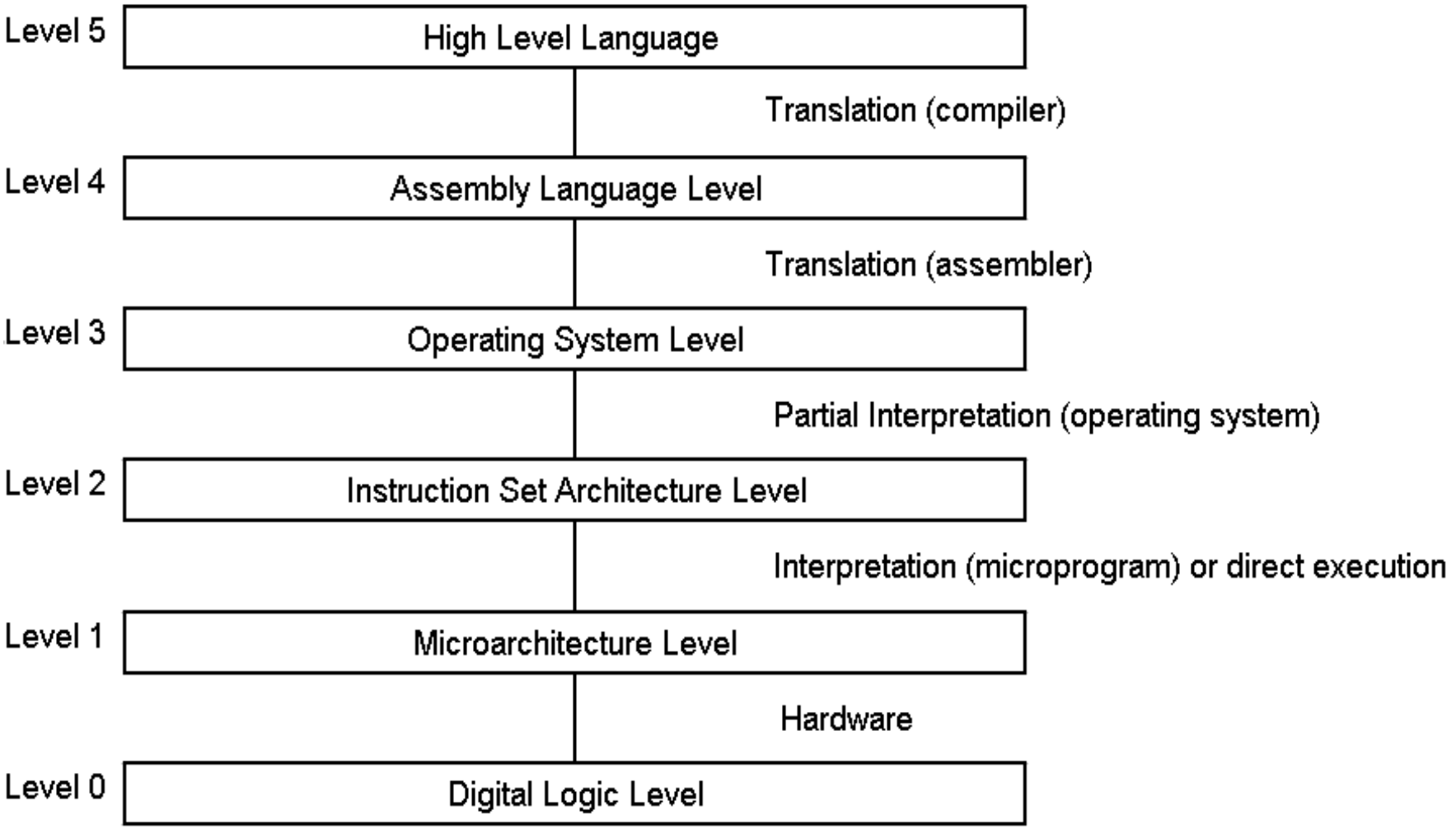
- Another Way of thinking



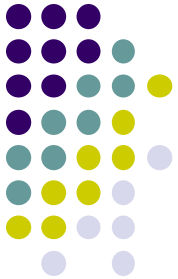
- A large gap between what is convenient for people and what is convenient for computers
- Solutions
  - Translation – entire program is translated into new language before being executed.
  - Interpretation – each instruction is examined and decoded, and then executed.
- Can create abstraction using virtual machines, each with it's own machine language
- Languages should not be too different

Virtual Machine Levels 0 through 5





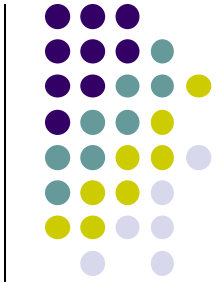
Multilevel Machines



- Level 0 – Digital Logic Level
  - Gate level, made from transistors, used to form 1-bit memories. Memories combined to form 16,32, or 64-bit registers.

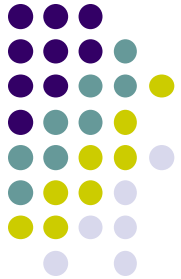
Level 1 – Microarchitecture Level

  - Collection of 8-32 registers that form a local memory and a circuit called an **ALU** (Arithmetic Logic Unit). Registers and ALU connected to form a **data path**. Data path may be controlled by software or hardware. Does fetching, examining, and executing of instructions, either through an interpreter or hardwired control.



Multilevel Machines

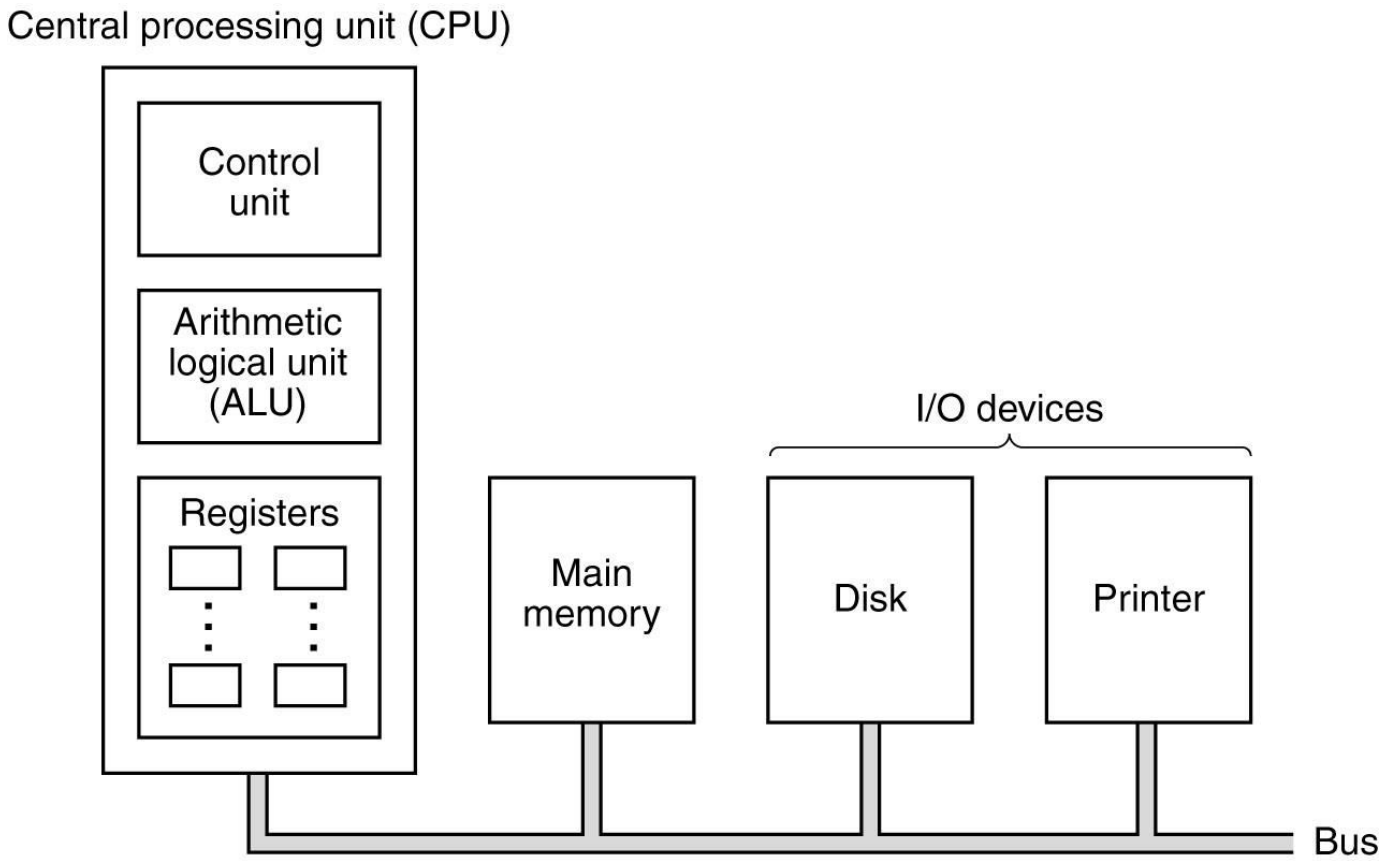
- Level 2 – Instruction Set Architecture
  - The instructions that are carried out interpretively by the underlying levels. Everything lower is proprietary.
- Level 3 – Operating System Machine Level
  - A new set of instructions (as well as those from level 2), a different memory organization, the ability to run 2 or more programs concurrently, various other features. Hybrid level.



Multilevel Machines

- Level 4 – Assembly Language Level
  - Finally have words and abbreviations rahter than just numbers. A symbolic form for one of the underlying languages.
- Level 5 – High-level Language Level
  - Languages designed to used by applications programmers.

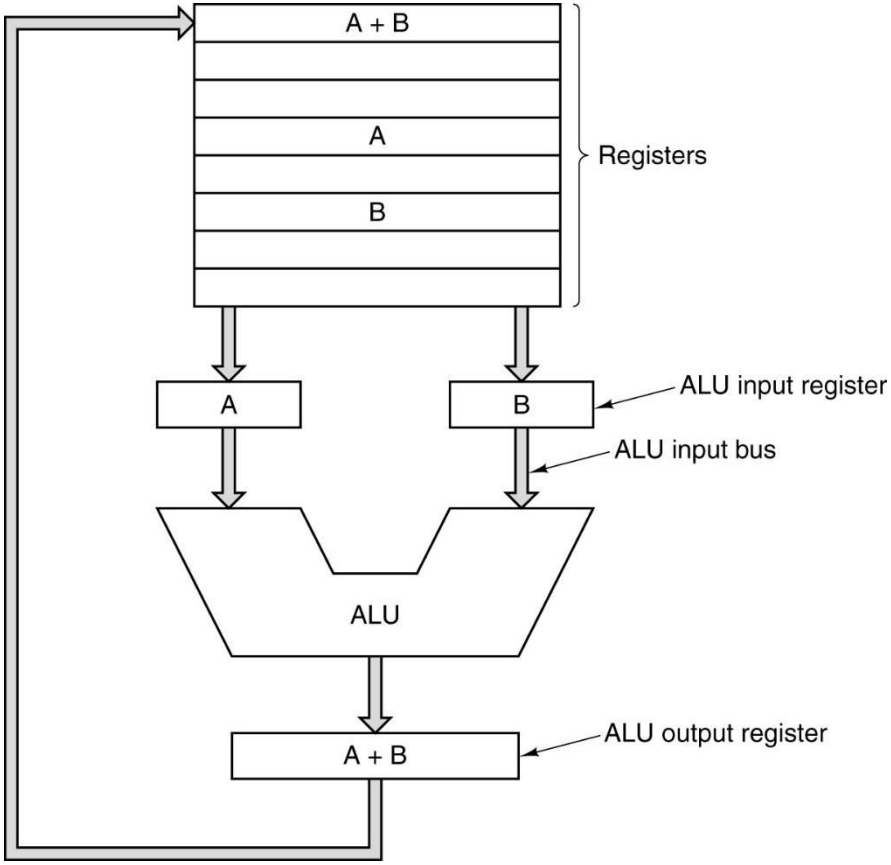
Central Processing Unit

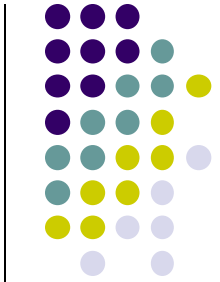


The organization of a simple computer with one CPU and two I/O devices

CPU Organization

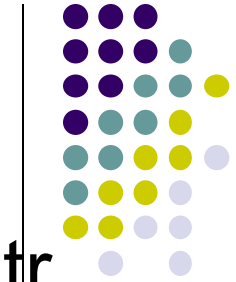
The data path of a typical Von Neumann machine.





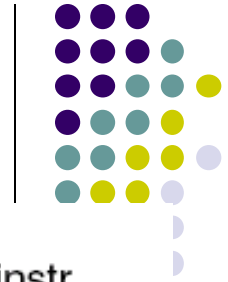
- Register- Memory Instruction
- Register-Register Instruction

Instruction Execution Steps



1. Fetch next instruction from memory into instr. register
2. Change program counter to point to next instruction
3. Determine type of instruction just fetched
4. If instructions uses word in memory, determine where Fetch word, if needed, into CPU register
5. Execute the instruction
6. Go to step 1 to begin executing following instruction

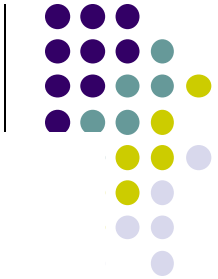
Interpreter (1)



```
public class Interp {
   static int PC;                    // program counter holds address of next instr
   static int AC;                    // the accumulator, a register for doing arithmetic
   static int instr;                 // a holding register for the current instruction
   static int instr_type;            // the instruction type (opcode)
   static int data_loc;              // the address of the data, or –1 if none
   static int data;                  // holds the current operand
   static boolean run_bit = true;    // a bit that can be turned off to halt the machine

   public static void interpret(int memory[ ], int starting_address) {
      // This procedure interprets programs for a simple machine with instructions having
      // one memory operand. The machine has a register AC (accumulator), used for
      // arithmetic. The ADD instruction adds an integer in memory to the AC, for example.
      // The interpreter keeps running until the run bit is turned off by the HALT instruction.
      // The state of a process running on this machine consists of the memory, the
      // program counter, the run bit, and the AC. The input parameters consist of
      // of the memory image and the starting address.
```

An interpreter for a simple computer (written in Java).

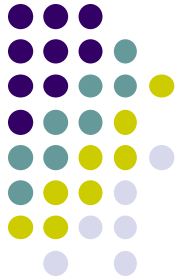
Interpreter (2)



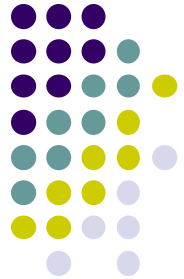
```
        PC = starting_address;
        while (run_bit) {
            instr = memory[PC];                    // fetch next instruction into instr
            PC = PC + 1;                           // increment program counter
            instr_type = get_instr_type(instr);    // determine instruction type
            data_loc = find_data(instr, instr_type); // locate data (–1 if none)
            if (data_loc >= 0)                     // if data_loc is –1, there is no operand
                data = memory[data_loc];           // fetch the data
            execute(instr_type, data);             // execute instruction
        }

    }

    private static int get_instr_type(int addr) { ... }
    private static int find_data(int instr, int type) { ... }
    private static void execute(int type, int data) { ... }
}
```

An interpreter for a simple computer (written in Java).

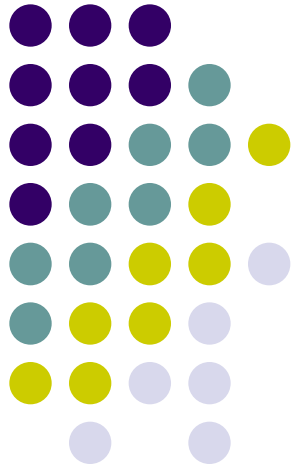
Design Principles for Modern Computers



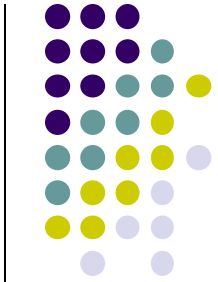
- All instructions directly executed by hardware
- Maximize rate at which instructions are issued
- Instructions should be easy to decode
- Only loads, stores should reference memory
- Provide plenty of registers



Performance



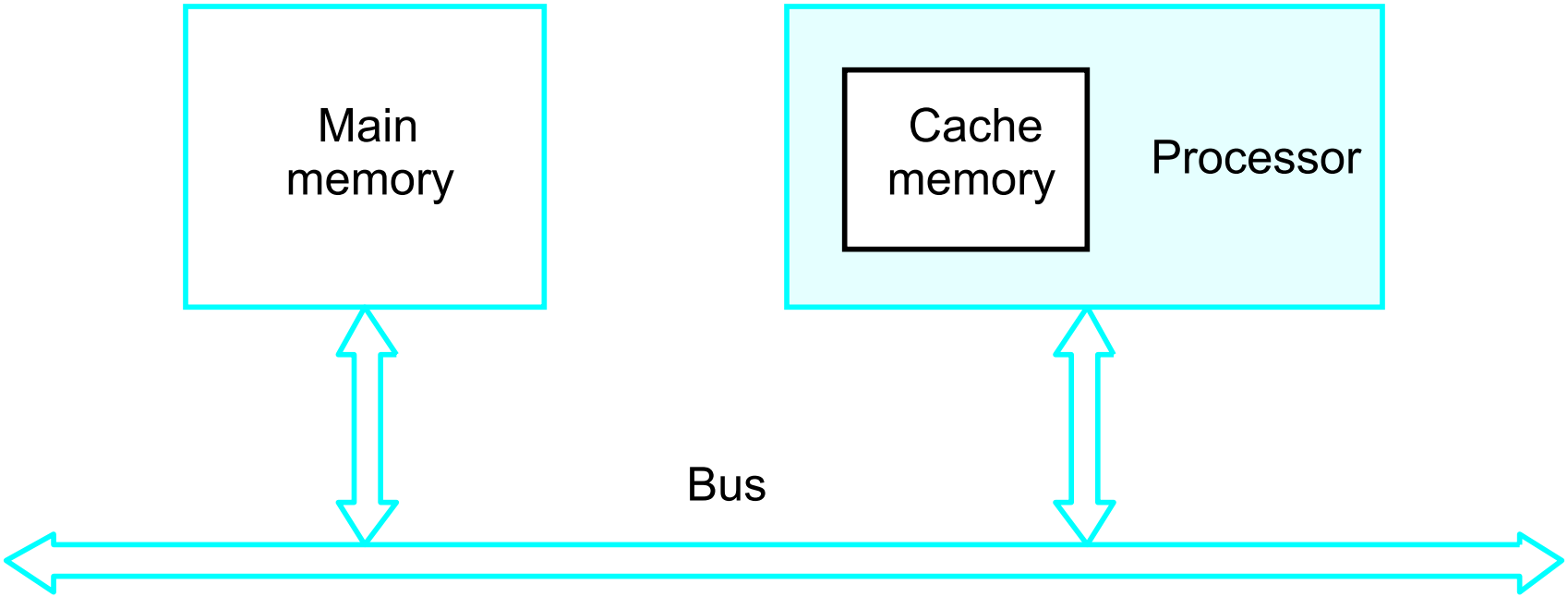
Performance



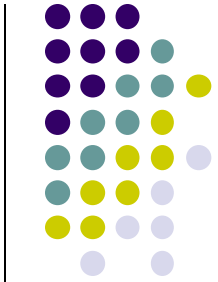
- The most important measure of a computer is how quickly it can execute programs.
- Three factors affect performance:
  - Hardware design
  - Instruction set
  - Compiler

Performance

- Processor time to execute a program depends on the hardware involved in the execution of individual machine instructions.



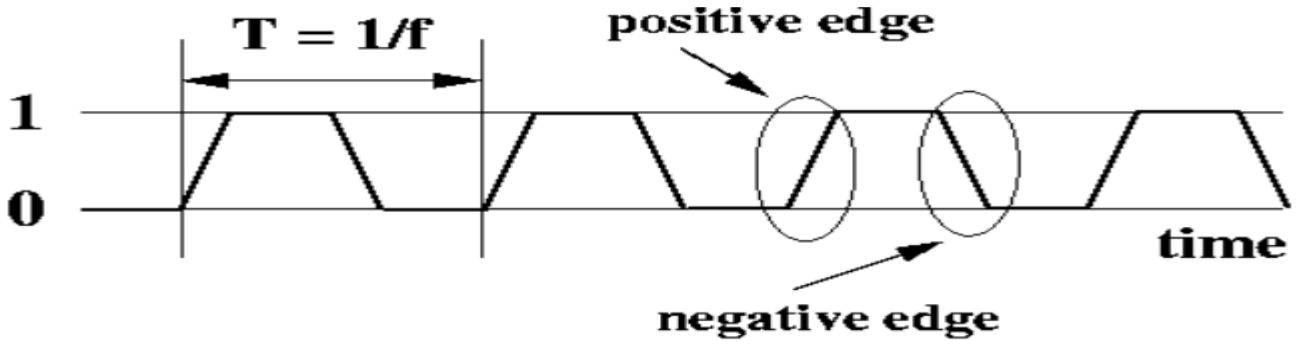
Performance



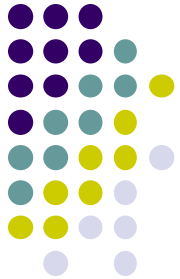
- The processor and a relatively small cache memory can be fabricated on a single integrated circuit chip.
- Speed
- Cost
- Memory management

Processor Clock

- Clock, clock cycle, and clock rate
- The execution of each instruction is divided into several steps, each of which completes in one clock cycle.
- Clock Rate= 1/P (Cycles per second)
- Hertz – cycles per second
- Million means Mega and billion means Giga (500 Million cycle per second is 500 MHz and 1250 million cycle per second is 1.25 GHz)



Basic Performance Equation

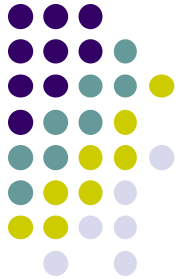


- T – processor time required to execute a program that has been prepared in high-level language
- N – number of actual machine language instructions needed to complete the execution (note: loop)
- S – average number of basic steps needed to execute one machine instruction. Each step completes in one clock cycle
- R – clock rate
- Note: these are not independent to each other

$$T = \frac{N \times S}{R}$$

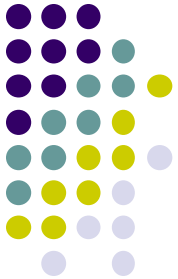
How to improve T?

Pipeline and Superscalar Operation



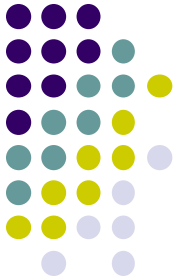
- Instructions are not necessarily executed one after another.
- The value of S doesn't have to be the number of clock cycles to execute one instruction.
- Pipelining – overlapping the execution of successive instructions.
- Add R1, R2, R3
- Superscalar operation – multiple instruction pipelines are implemented in the processor.
- Goal – reduce S (could become <1!)

Clock Rate



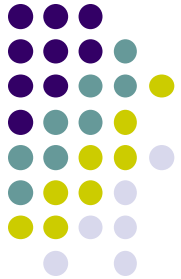
- Increase clock rate
  - Improve the integrated-circuit (IC) technology to make the circuits faster
  - Reduce the amount of processing done in one basic step (however, this may increase the number of basic steps needed)
- Increases in R that are entirely caused by improvements in IC technology affect all aspects of the processor's operation equally except the time to access the main memory.

CISC and RISC



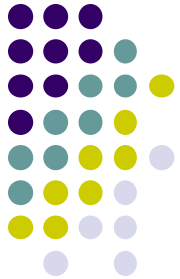
- Tradeoff between N and S
- A key consideration is the use of pipelining
  - S is close to 1 even though the number of basic steps per instruction may be considerably larger
  - It is much easier to implement efficient pipelining in processor with simple instruction sets
- Reduced Instruction Set Computers (RISC)
- Complex Instruction Set Computers (CISC)

Compiler



- A compiler translates a high-level language program into a sequence of machine instructions.
- To reduce N, we need a suitable machine instruction set and a compiler that makes good use of it.
- Goal – reduce N×S
- A compiler may not be designed for a specific processor; however, a high-quality compiler is usually designed for, and with, a specific processor.

Performance Measurement

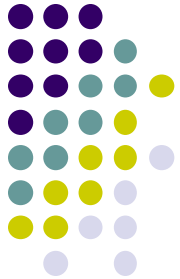


- T is difficult to compute.
- Measure computer performance using benchmark programs.
- System Performance Evaluation Corporation (SPEC) selects and publishes representative application programs for different application domains, together with test results for many commercially available computers.
- Compile and run (no simulation)
- Reference computer

$$SPEC\ rating = \frac{\text{Running time on the reference computer}}{\text{Running time on the computer under test}}$$

$$SPEC\ rating = \left(\prod_{i=1}^{n} SPEC_i\right)^{\frac{1}{n}}$$

Multiprocessors and Multicomputers



- Multiprocessor computer
  - Execute a number of different application tasks in parallel
  - Execute subtasks of a single large task in parallel
  - All processors have access to all of the memory – shared-memory multiprocessor
  - Cost – processors, memory units, complex interconnection networks
- Multicomputers
  - Each computer only have access to its own memory
  - Exchange message via a communication network – message-passing multicomputers

Steps needed to execute the machine instruction

- Load R2, LOC

Send the address of the instruction word from register PC to the memory and issue a Read control command.

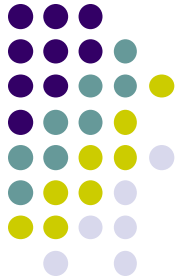
Wait until the requested word has been retrieved from the memory, then load it into register IR, where it is interpreted (decoded) by the control circuitry to determine the operation to be performed.

Increment the contents of register PC to point to the next instruction in memory.

Send the address value LOC from the instruction in register IR to the memory and issue a Read control command.

Wait until the requested word has been retrieved from the memory, then load it into register R2.

Question



(a)Give a short sequence of machine instructions for the task “Add the contents of memory location A to those of location B, and place the answer in location C”. Instructions

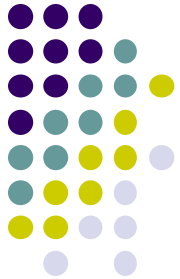
Load R*i*, LOC

and

Store R*i*, LOC

are the only instructions available to transfer data between the memory and the general purpose registers.

Add instructions are described “Add LOCA, R0”. Do not change the contents of either location A or B.



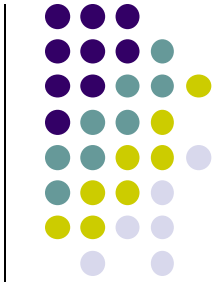
(a)

Load A,R0

Load B,R1

Add R0,R1

Store R1,C



(*b*) Suppose that Move and Add instructions are available with the formats

Move Location1, Location2

and

Add Location1, Location2

These instructions move or add a copy of the operand at the second location to the first location, overwriting the original operand at the first location. Either or both of the operands can be in the memory or the general-purpose registers. Is it possible to use fewer instructions of these types to accomplish the task in part (*a*)? If yes, give the sequence.