Unit-1
Basic Structure of computers

a) Computer Types
b) Functional Units
c) Basic Operational Concepts
d) Bus structure
e) Performance- Processor Clock, Basic Performance Equation, Clock rate
f) Performance Measurement
g) Historical Perspective
h) Machine Instructions and Programs:
   i. Numbers, Arithmetic Operations and Characters
   ii. Memory Location and Addresses
   iii. Memory Operations
   iv. Instructions and Instruction Sequencing

Computer Types

1. Mainframe
2. Minicomputer
3. Supercomputer
4. Desktop computer
5. Server
6. Embedded computer

1. Mainframe

- Year 1960
- Costly
- Large size
- Multi-user
- Application: Data Processing & Scientific Computing (Bank, Government, Corporate)
- Response 100 sec. for million user

2. Minicomputer

- Year 1970
- Costlier
- Small size
- Multi-user
- Application: Data Processing & Scientific Computing (Scientific Laboratory)
- Time-sharing

3. Supercomputer

- Year 1970
- Very large computer in the form of memory space & operating speed
- Use in number crunching and simulation studies in bigger organization (like, business houses or universities)

4. Desktop Computer

- Year 1980
- Less Costlier
- Small size
- Feature: Microprocessor
- Two Classes:
  - » Personal Computer also called Micro-computer

    Alternate of timesharing minicomputer, flexible and meet a wide range of end user needs
  - » Workstation: single user and contain special hardware

5. Server

- Year 1980
- Costlier
- Size
- Dedicated to provide large scale services
  - Reliable
  - Long-term file storage and access
  - Large memory
  - More computing power

- Personal Digital Assistant
  - Year 1990
  - First hand held computing devices
  - High performance digital consumer electronics
    video game, set-top box

6. Embedded Computer
   - Year 2000
   - Controlled by microcontrollers, a single-chip computer and contain all necessary hardware and software to run the system for which it is programmed.
   - Not capable of executing any user program
   - Handle particular task
   - Battery or solar cell operated, consume extremely less power
   - Reduce the size and product cost
   - ipods, digital camera, cellphone, digital watches, mp3 player, factory controller, automatic weighing machines, robots

Functional Units

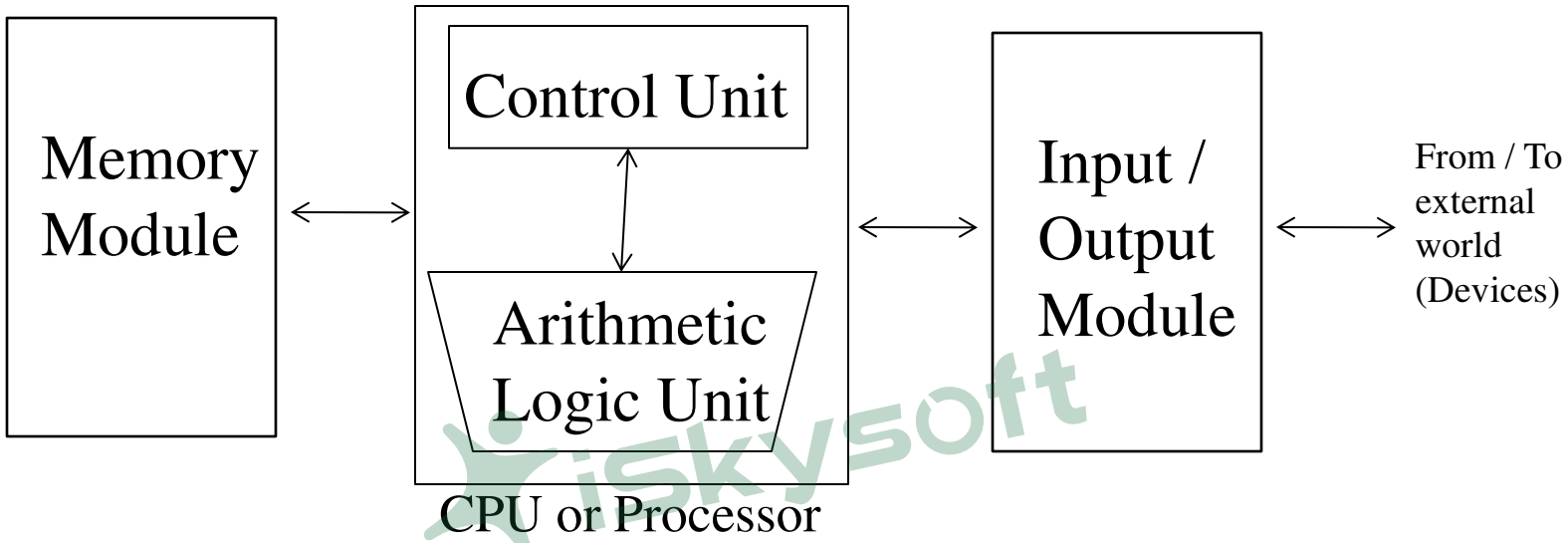
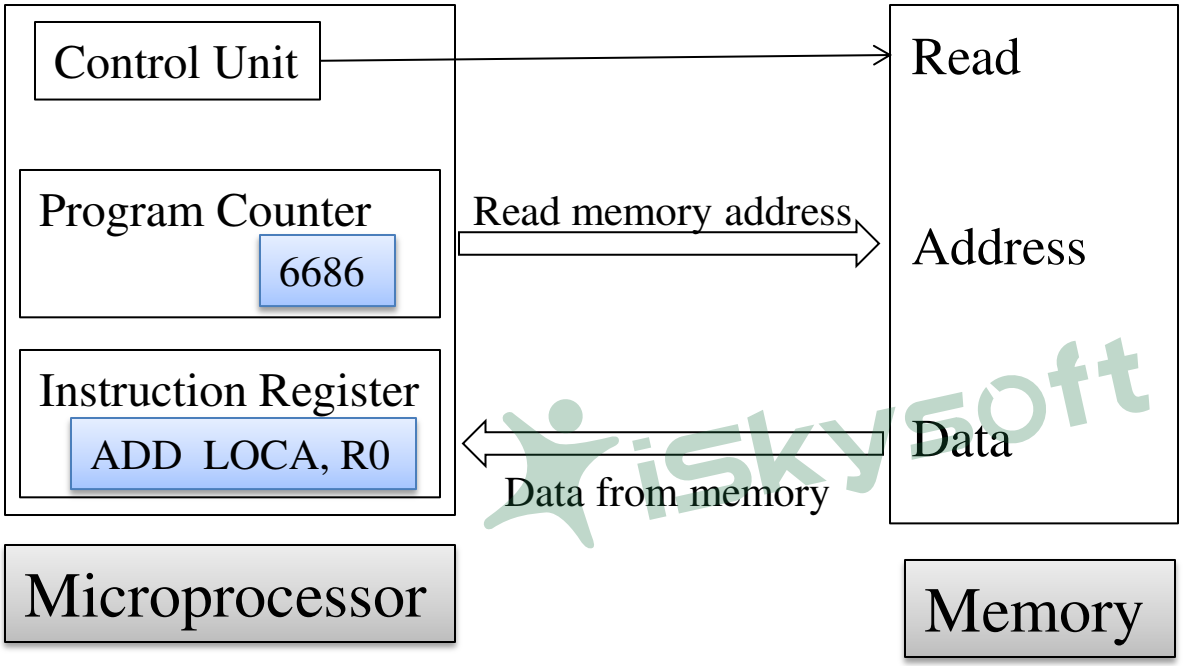


Fig: Schematic representation of a computer

Reading from memory

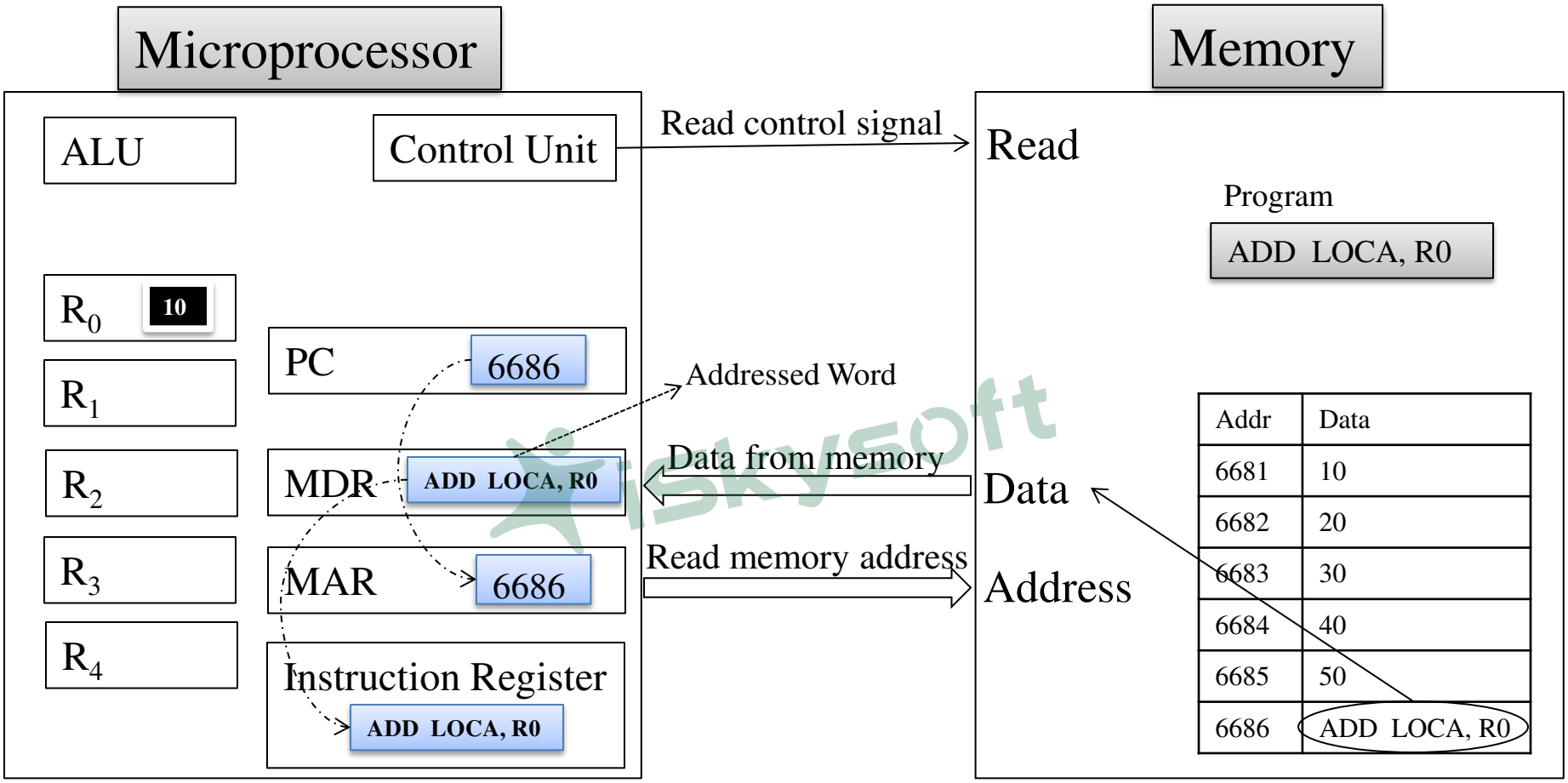
Program

ADD LOCA, R0

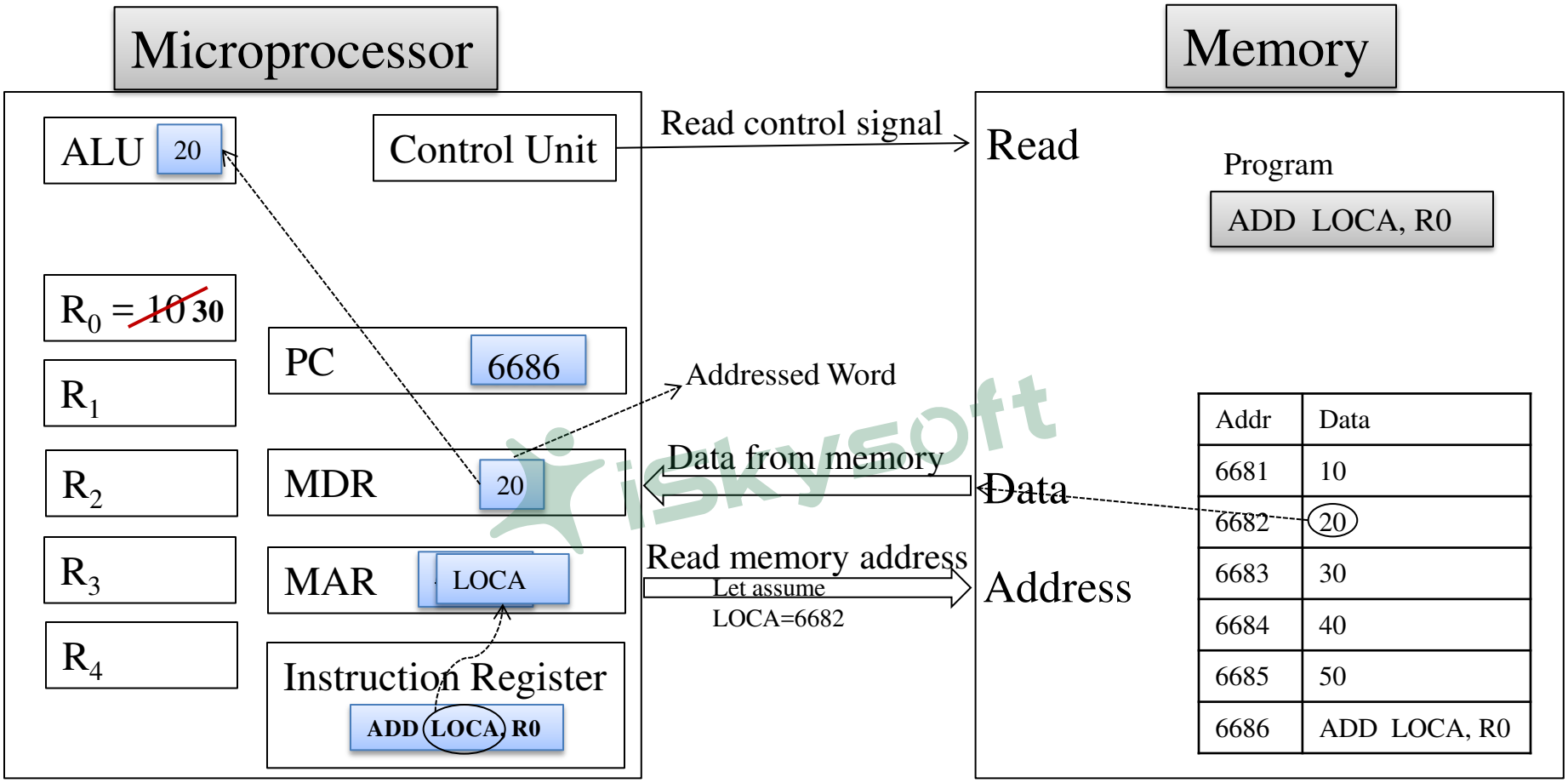


Addr	Data
6681	10
6682	20
6683	30
6684	40
6685	50
6686	ADD LOCA, R0

Basic Operational Concepts



Basic Operational Concepts



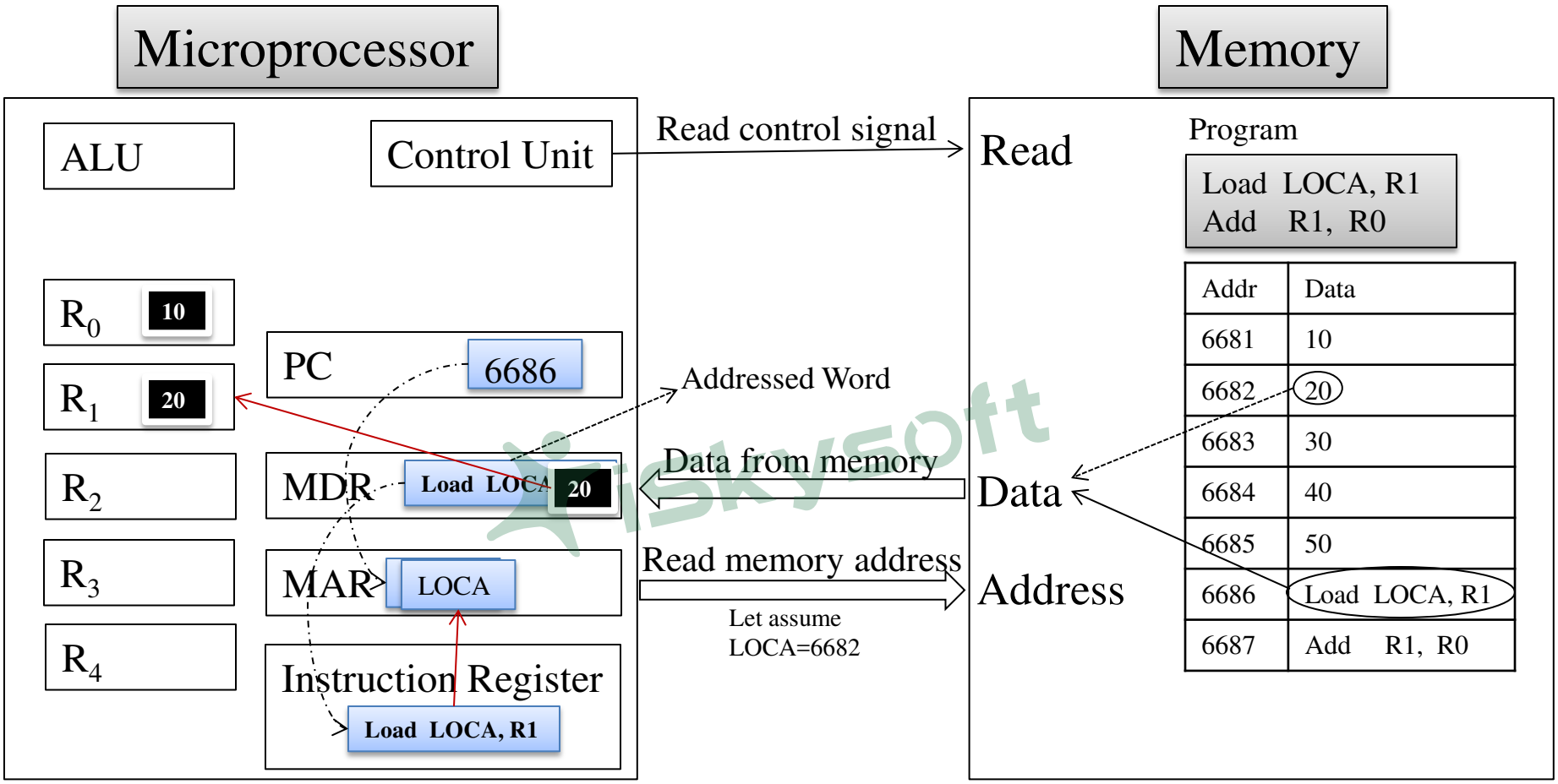
Program

```
ADD  LOCA, R0
```

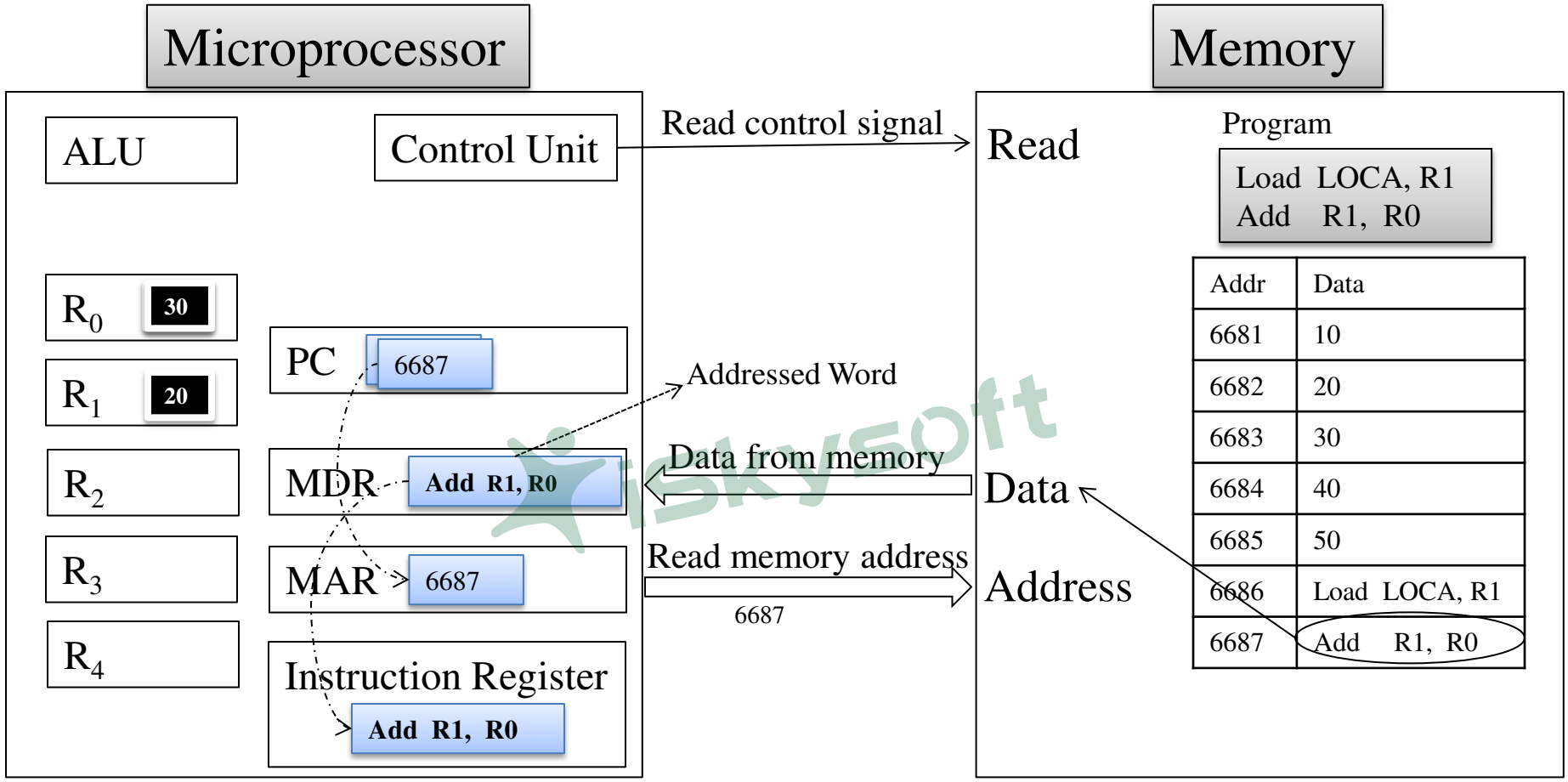
Basic Operational Steps for Execution of ADD Instruction

1. Program Counter (PC) shows a memory address of first instruction of the Program i.e. 6686.
2. The content of PC is transferred to the MAR.
3. The Control Unit sends the read control signal to the memory.
4. The MAR holds the address 6686 of the location to be accessed. The address signal is emitted from the MAR to the memory with the help of the Address Bus, to target the desired memory location.
5. After targeting the desired memory location, Next it has to be brought inside the processor on MDR through Data Bus.
6. The contents of the MDR are transferred to the Instruction Register (IR).
7. The IR holds the ADD instruction. One operand (R0) is available while the second operands resides in the memory at location LOCA.
8. IR sends LOCA (i.e. 6682) to MAR.
9. The content of address LOCA is targeted in the memory and fetched into the MDR from memory for read operation.
10. The contents (20) of the MDR is shifted into the ALU.
11. After obtaining the operands (20), the ALU adds this operands (20) to the operand in a register R0 (10) in the processor, and places the sum into register R0.
12. The Original contents of LOCA (6682) is preserved whereas the original contents (10) of R0 is overwritten by sum (30).

Basic Operational Concepts

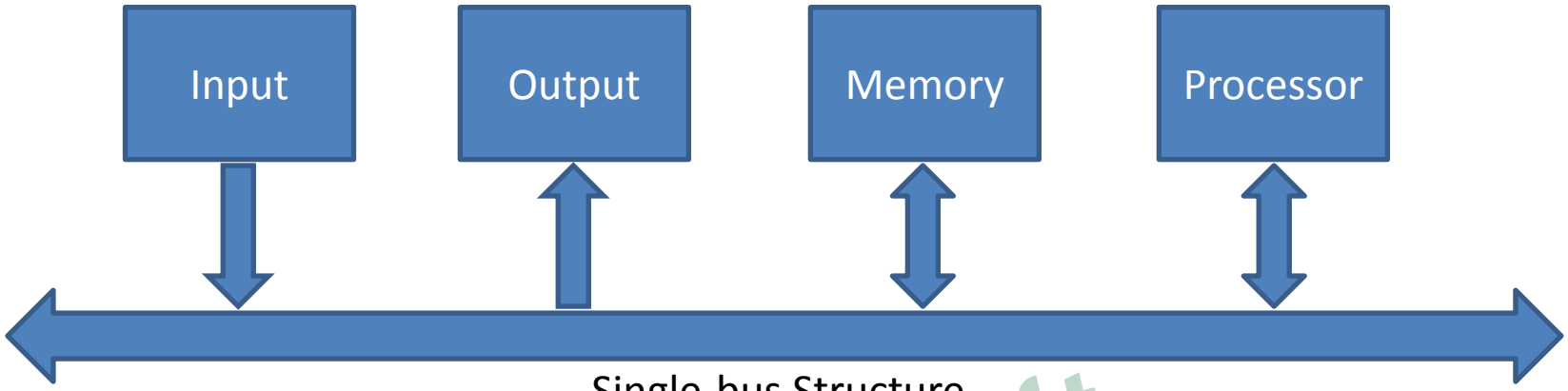


Basic Operational Concepts



Bus Structures

Bus is a bunch of signal lines intended to carry digital signals in computer systems



Single-bus Structure

Two Types of buses:

1) Internal bus → Synchronous communication techniques
   a) Address bus → Uni-directional
   b) Data bus → Bi-directional
   c) Control bus → Uni-directional

2) External bus → Asynchronous communication techniques
   - Handshaking signal

Performance

What is the duty of any computer when it is operational?

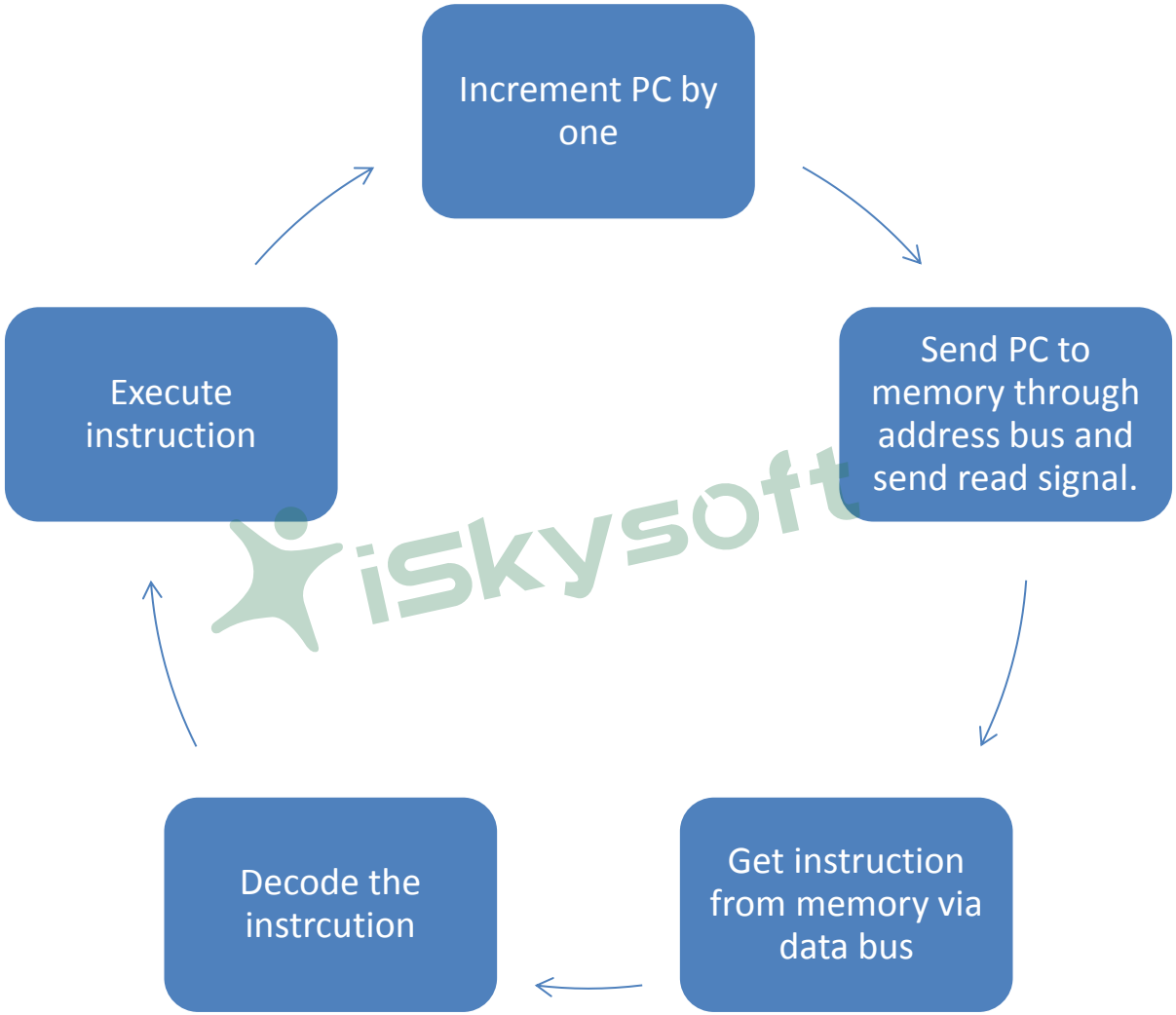
1. Execute the software by fetching instructions from memory.
2. Look for any external signal and react accordingly.

Major duty of the processor is to run (or execute) the software, and this program execution is done continuously until the computer is switched off.

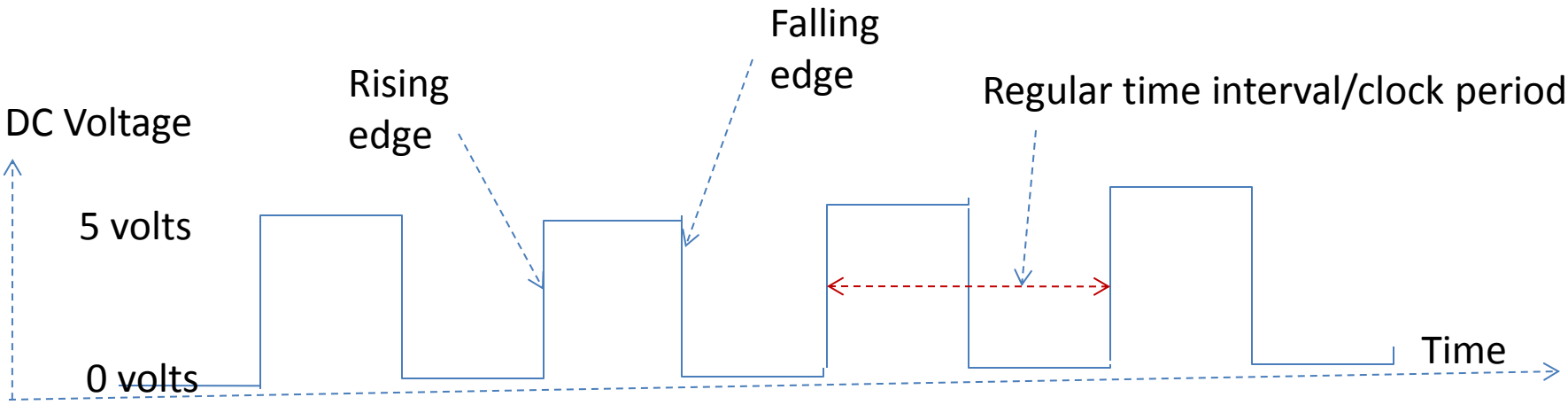
How this execution is done?

Several steps are to be combined together to respond to this question.

Major functions of a processor



Processor clock



The heart of any processor is its clock, which starts almost the moment a computer is switched on (powered). This clock is a simple digital signal producing ON or OFF states alternately, at equal time intervals.

Which counter is affected by this process?
The answer is –the Program Counter.

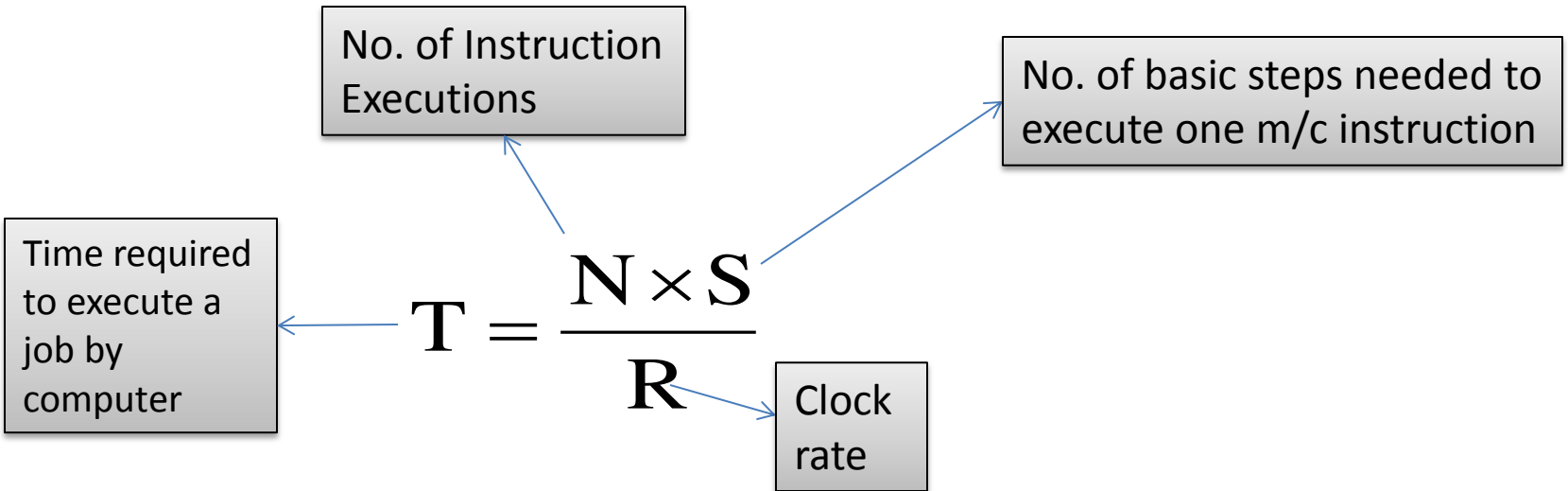
length of one clock cycle = P
Clock rate, $R = \frac{1}{P}$ cycles per second or hertz
Million→ Mega (M)
Billion →Giga (G)

Clock Rate R	Clock period P
500 MHz	2 ns
1.25 GHz	0.8 ns

Basic Performance Equation

User	Laboratory Engineer
Time taken to execute a job (program)	Total amount of work done in a given time
Program execution time a measure for performance	Throughput a measure for performance

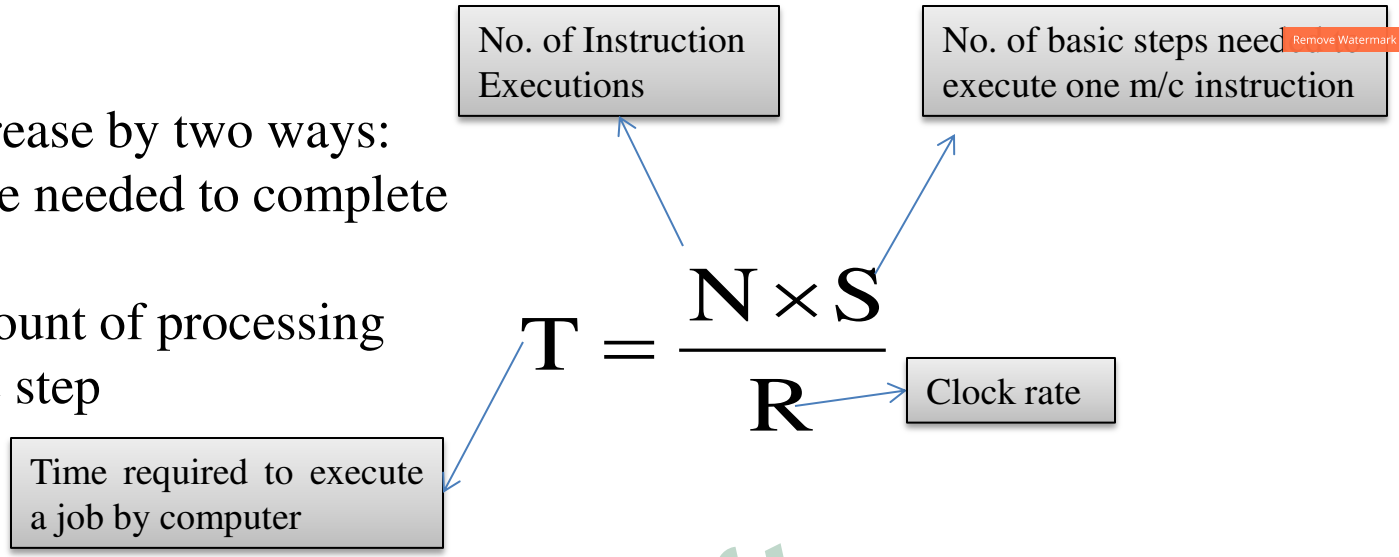
Performance analysis should help answering questions such as how fast can a program be executed using a given computer?
In order to answer such a question, we need to determine the time taken by a computer to execute a given job.



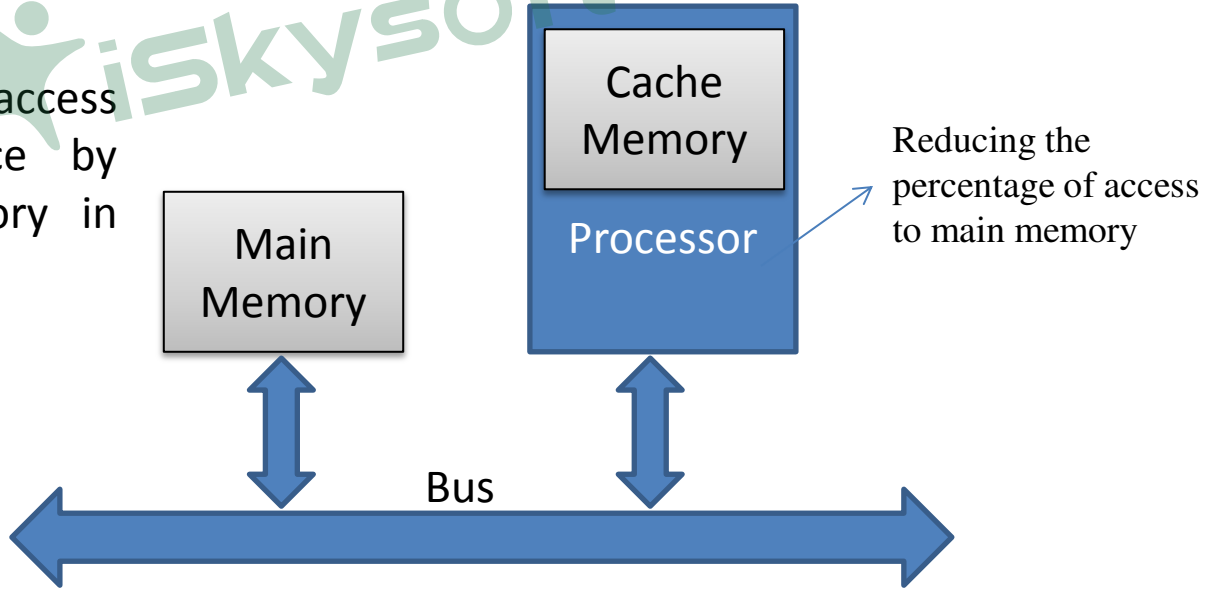
Clock Rate

Clock rate can be increase by two ways:

1. Reducing the time needed to complete a basic step
2. Reducing the amount of processing done in one basic step



The Time processor takes to access main memory can be reduce by maintaining small cache memory in side the processor.



The Processor cache

Performance Measurement

SPEC: stands for System performance Evaluation Corporation

It selects and publishes representative application programs for different application domains, together with test results for many commercially available computers

The benchmark program is compiled and run on one computer selected as a reference.

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

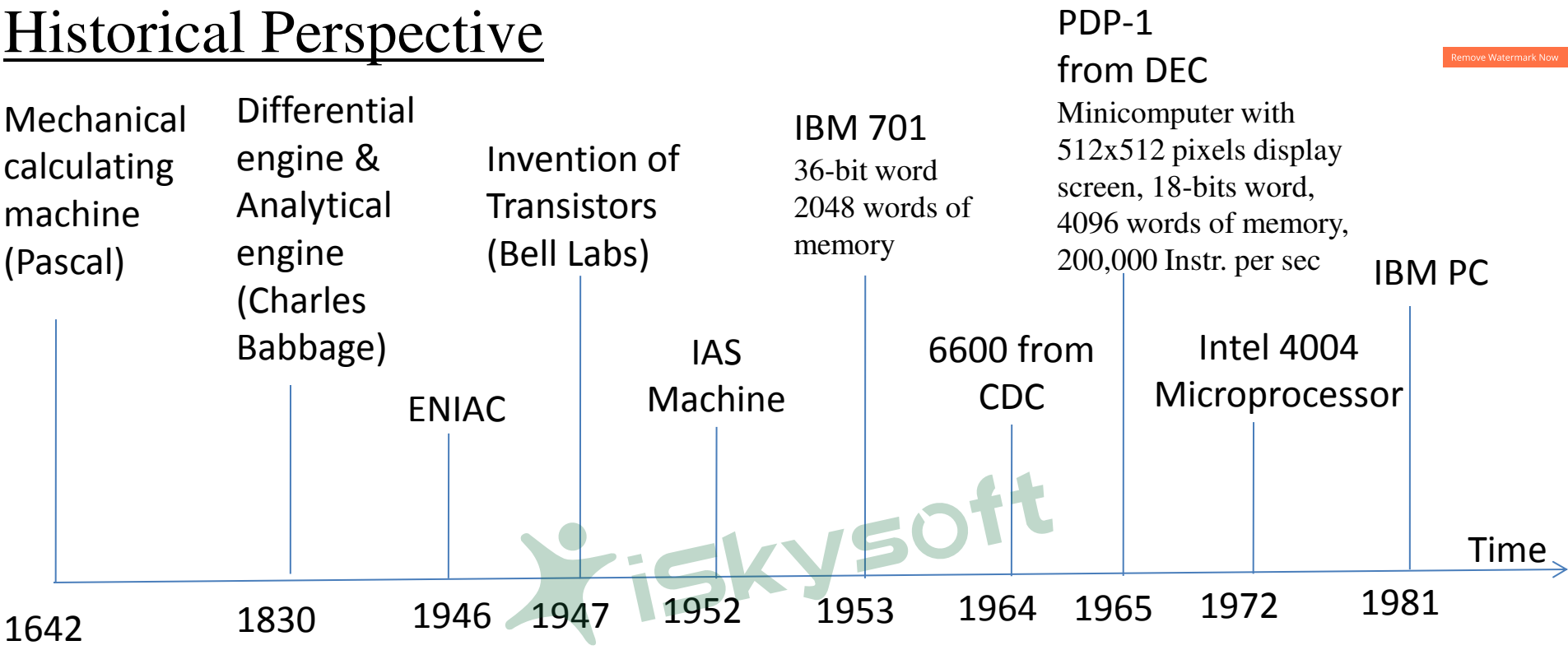
$$\text{SPEC rating} = \left(\prod_{i=1}^{n} \text{SPEC}_i \right)^{\frac{1}{n}}$$

Benchmark Program	Reference Computer
SPEC95	SUN SPARCstation 10/40
SPEC2000	Ultra-SPARC 10 workstation with 300-MHz UltraSPARC-Iii processor

SPEC suit has collection of programs ranging from game playing, compiler, database applications to numerically intensive programs in astrophysics and quantum chemistry

SPEC rating: It is a measure of the combined effect of all factors affecting performance, including the compiler, the operating system, the processor, and memory of the computer.

Historical Perspective



Time-line of Improvements in Computer

ENIAC: Electronic numerical integrator and computer
IAS Machine: Institute for advanced study
CDC: control data corporation
PDP: Programmed data processor
DEC: Digital Equipment corporation

Historical Perspective

Generations of Computers	Technology & Application	Software and Examples	Popular Model of Computers
First Generation (1945-54)	Vacuum tubes & relay memories, CPU driver by PC and Accumulator Fixed-point Arithmetic	Machine/Assembly languages, single user, no subroutine linkage, programmed I/O using CPU	ENIAC, IAS Machine, EDVAC, IBM 650, IBM 701
Second Generation (1955-64)	Transistors, core memories, I/O processor, Floating-point calculators	High-level language used with compilers, subroutine libraries, Batch processing monitor	ATLAS, B-5000, PDP-1, IBM 7090, CDC 1604

Historical Perspective

Generations of Computers	Technology & Application	Software and Examples	Popular Model of Computers
Third Generation (1965-74)	Integrated circuits (SSI/MSI), microprogramming, pipelining, cache and look ahead processors	Multiprogramming and time sharing Operating systems	IBM 360/370, CDC 6600, TI-ASC, HP 2100 PDP-8
Fourth Generation (1975-90)	LSI/VLSI and Semiconductor memory multiprocessors, vector supercomputers, multicomputer	Multiprocessor OS, Languages, Compiler and Environments for parallel processing	VAX 9000, Intel 8080, Cray XMP, IBM 3090

Historical Perspective

Generations of Computers	Technology & Application	Software and Examples	Popular Model of Computers
Fifth Generation (1991-Till)	ULSI/VHSIC processors, memory and switches High density packaging scalable architectures	Massively parallel processing, grand challenge Applications, heterogeneous processing	IBM RS-16000, Fujitsu VPP 500, Intel Paragon

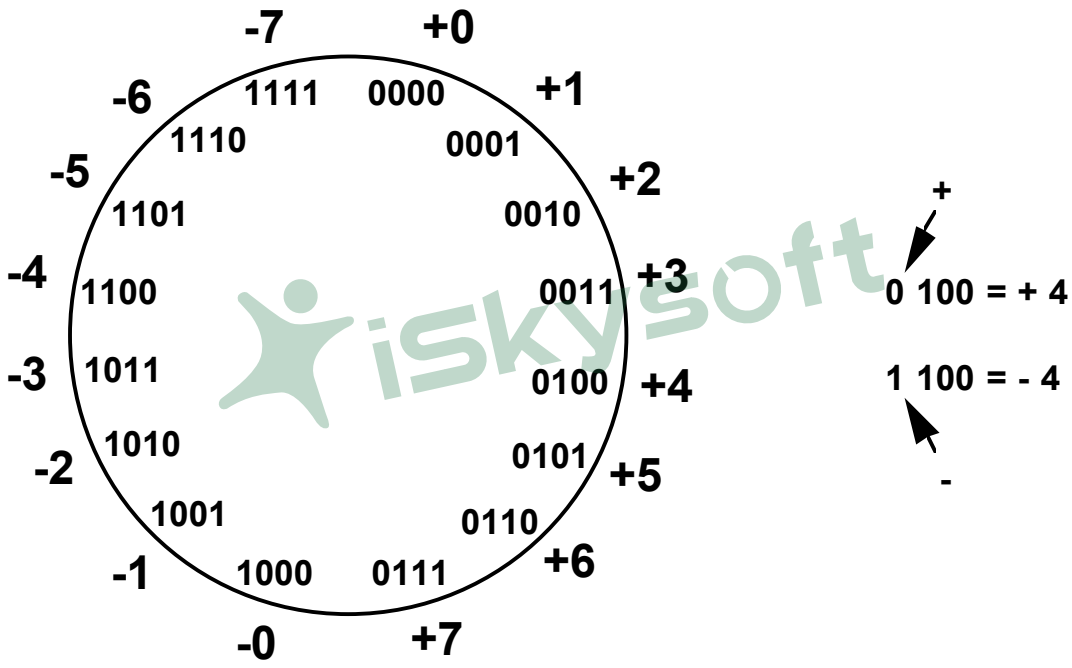
Number, Arithmetic Operations, and Characters

Number representation

Signed Integer

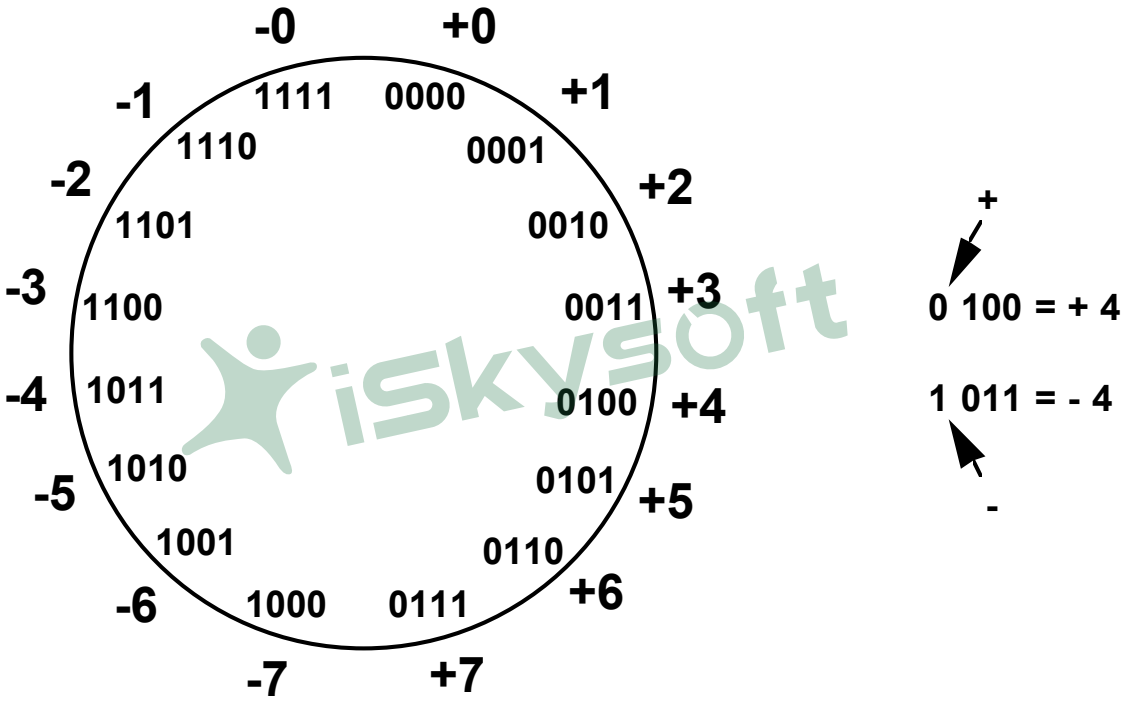
- 3 major representations:
  - Sign and magnitude
  - One's complement
  - Two's complement
- Assumptions:
  - 4-bit machine word
  - 16 different values can be represented
  - Roughly half are positive, half are negative

Sign and Magnitude Representation



High order bit is sign: 0 = positive (or zero), 1 = negative
Three low order bits is the magnitude: 0 (000) thru 7 (111)
Number range for n bits = +/-2^{n-1} -1
Two representations for 0

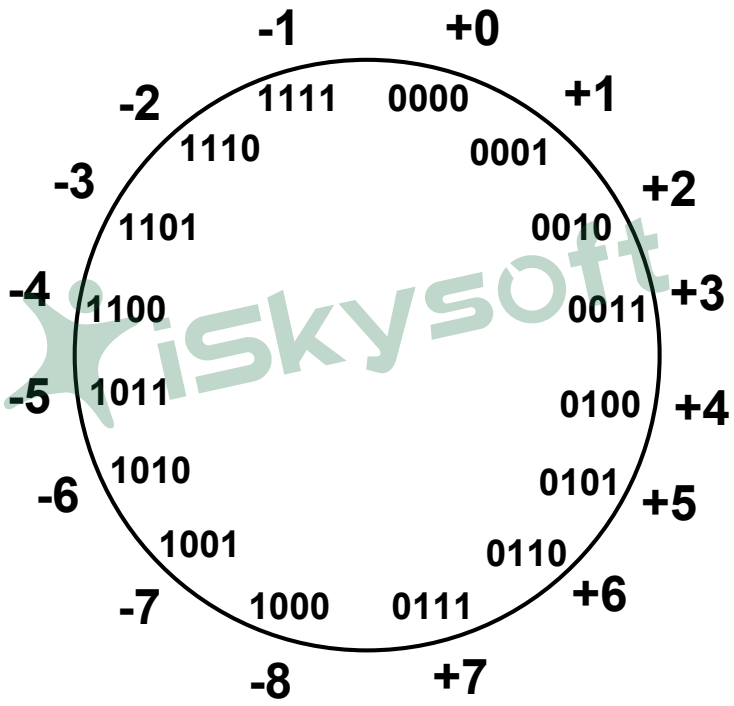
One’s Complement Representation

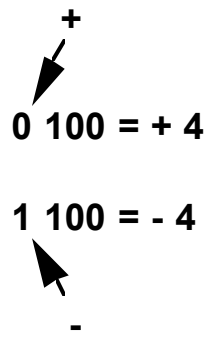


- Subtraction implemented by addition & 1's complement
- Still two representations of 0! This causes some problems
- Some complexities in addition

Two's Complement Representation

like 1's comp except shifted one position clockwise





- Only one representation for 0
- One more negative number than positive number

Two’s complement calculation

Case-I

Original binary number	1	0	1	0
One’s complement	0	1	0	1
Add one				1
Two’s complement	0	1	1	0

Case-III

Original binary number	1	1	1	1
One’s complement	0	0	0	0
Add one				1
Two’s complement	0	0	0	1

Case-II

Original binary number	1	0	0	0
One’s complement	0	1	1	1
Add one				1
Two’s complement	1	0	0	0

Same as the original number

Case-IV

Original binary number		0	0	0	0
One’s complement		1	1	1	1
Add one					1
Two’s complement	1	0	0	0	0

Carry

Signed-decimal number	Signed-magnitude representation	Two's complement representation
+7	0111	0111
+6	0110	0110
+5	0101	0101
+4	0100	0100
+3	0011	0011
+2	0010	0010
+1	0001	0001
+0	0000	0000
-0	1000	0000
-1	1001	1111
-2	1010	1110
-3	1011	1101
-4	1100	1100
-5	1101	1011
-6	1110	1010
-7	1111	1001

Signed-magnitude and two's complement representation of decimal numbers (within the range of +7 to -7, using 4-bit representation)

Binary, Signed-Integer Representations

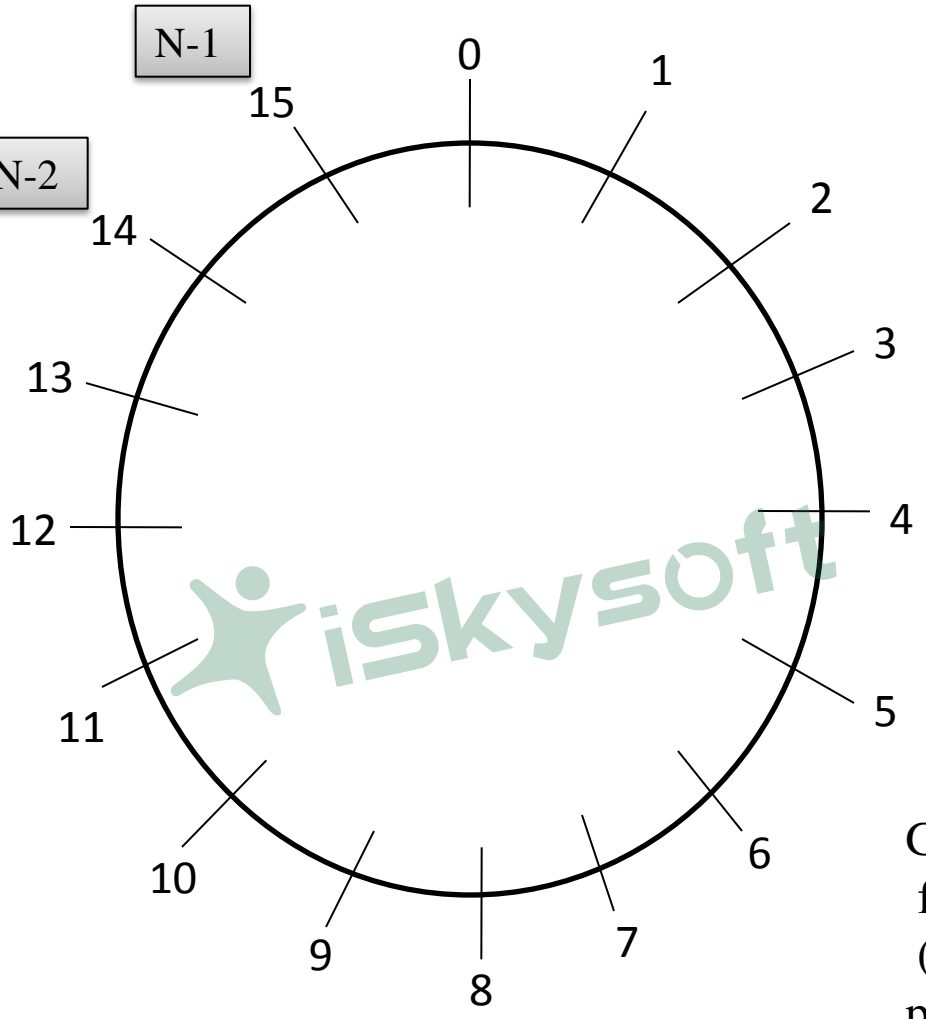
Page 28

B	Values represented		
$b_3 b_2 b_1 b_0$	Sign and magnitude	1's complement	2's complement
0 1 1 1	+ 7	+ 7	+ 7
0 1 1 0	+ 6	+ 6	+ 6
0 1 0 1	+ 5	+ 5	+ 5
0 1 0 0	+ 4	+ 4	+ 4
0 0 1 1	+ 3	+ 3	+ 3
0 0 1 0	+ 2	+ 2	+ 2
0 0 0 1	+ 1	+ 1	+ 1
0 0 0 0	+ 0	+ 0	+ 0
1 0 0 0	- 0	- 7	- 8
1 0 0 1	- 1	- 6	- 7
1 0 1 0	- 2	- 5	- 6
1 0 1 1	- 3	- 4	- 5
1 1 0 0	- 4	- 3	- 4
1 1 0 1	- 5	- 2	- 3
1 1 1 0	- 6	- 1	- 2
1 1 1 1	- 7	- 0	- 1

Figure 2.1. Binary, signed-integer representations.

Addition and Subtraction of Signed numbers

N=16



(7+4) mod 16
= (11) mod 16
=11

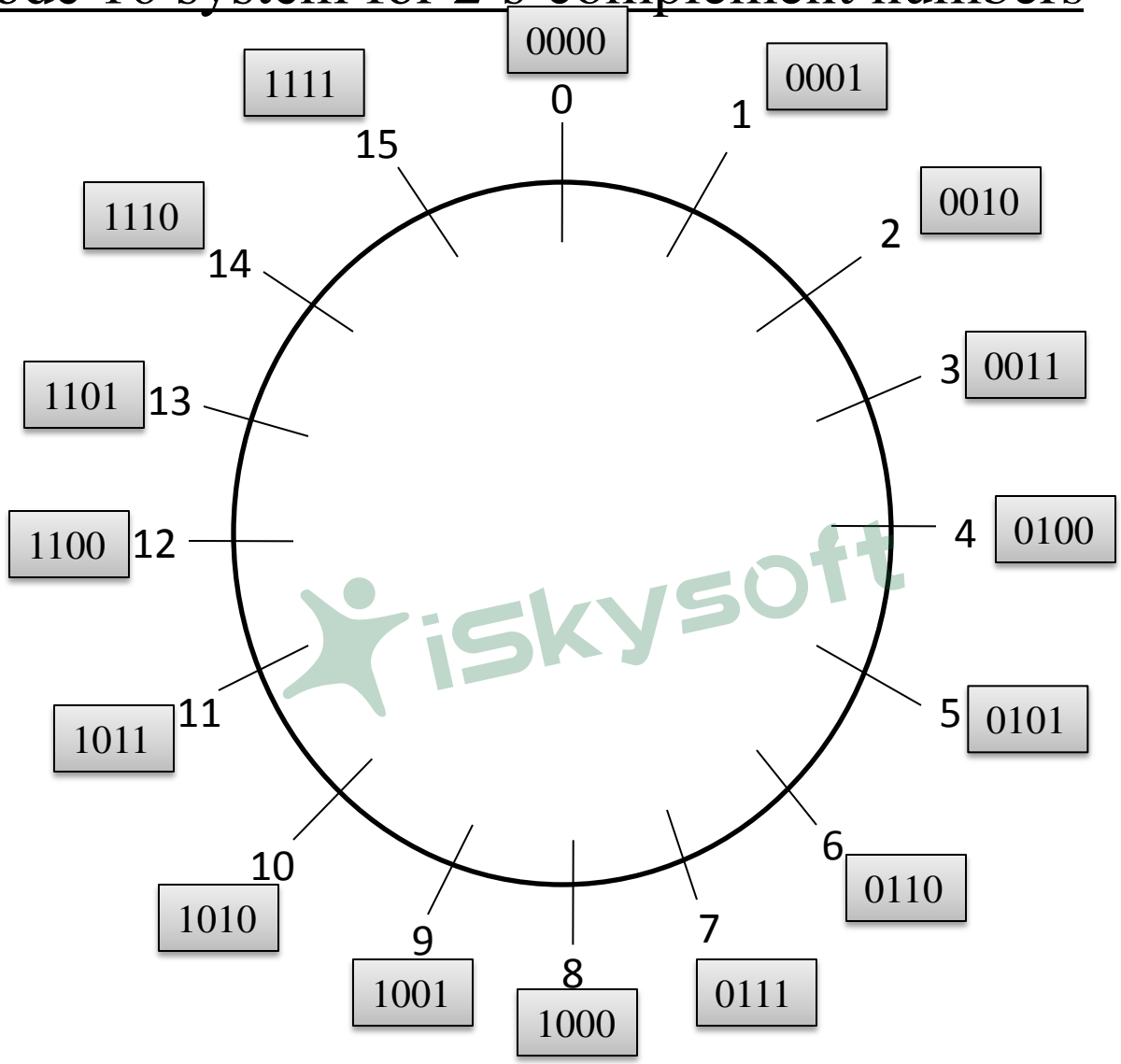
(9+14) mod 16
= (23) mod 16
=7

Graphical Techniques for computation of (a+b) mod 16 for any positive numbers a and b

Circle representation of positive integers mod N

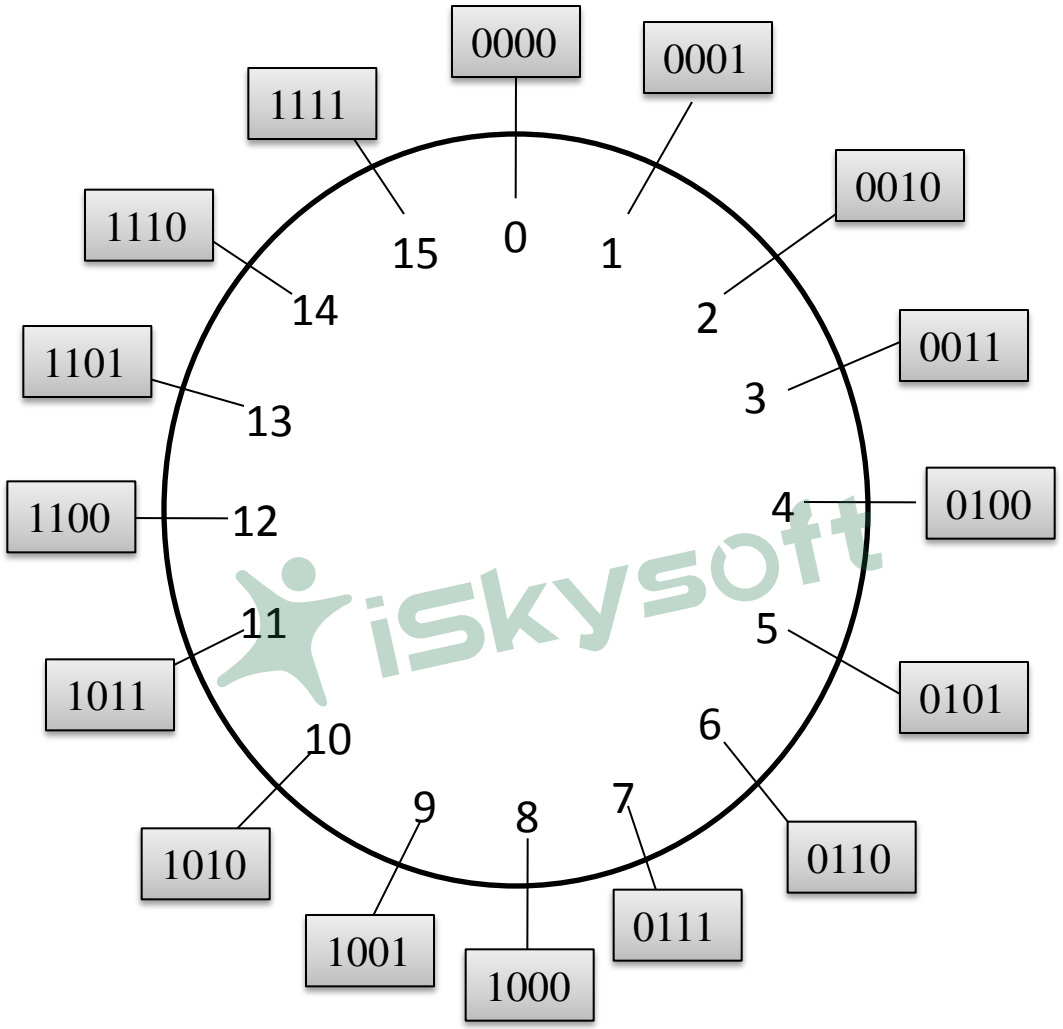
Modular number systems and the 2's complement system

Mode 16 system for 2's-complement numbers



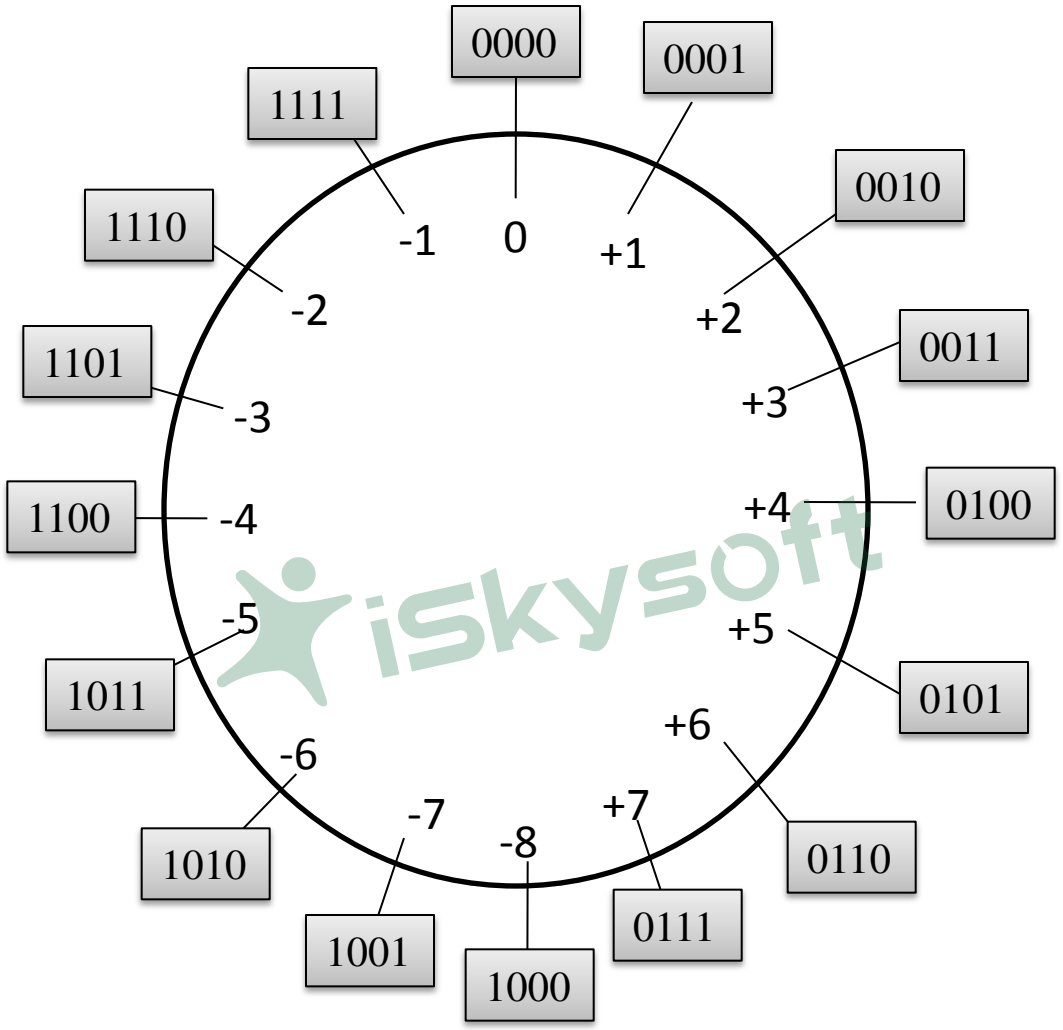
Circle representation of positive integers mod N

Mode 16 system for 2's-complement numbers



Circle representation of positive integers mod N

Mode 16 system for 2's-complement numbers



Reinterpret binary vectors to represent the signed numbers from -8 through +7 in the 2's complement method

Addition and Subtraction – 2's Complement

(+7)	0111
+ (-3)	1101
4	10100

Carry

4	0100
+ 3	0011
7	0111

-4	1100
+ (-3)	1101
-7	11001

Carry

4	0100
- 3	1101
1	10001

Carry

-4	1100
+ 3	0011
-1	1111

Simpler addition scheme makes two's complement the most common choice for integer number systems within digital systems

2’s-Complement Add and Subtract Operations

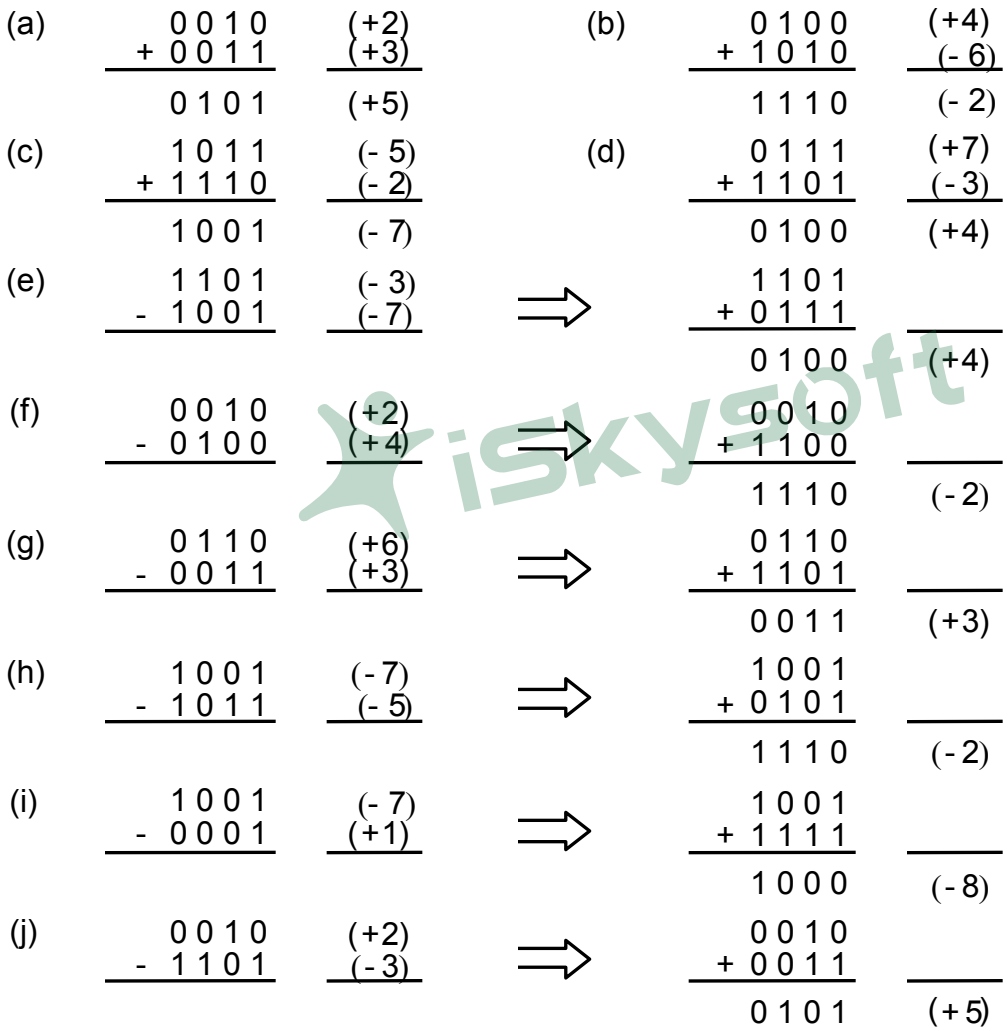


Figure 2.4. 2's-complement Add and Subtract operations.

Rules governing the addition and subtraction of n-bit signed numbers using the 2's-complement representation system

1. To *add* two numbers, add their n-bit representations, ignoring the carry-out signal from the *most significant bit* (MSB) position. The sum will be the algebraically correct value in the 2's-complement representation as long as the answer is in the range -2^{n-1} through $+2^{n-1}-1$

2. To *subtract* two numbers X and Y, that is, to perform X-Y, from the 2's complement of Y and then add it to X, as in rule 1. Again, the result will be the algebraically correct value in the 2's complement representation system if the answer is in the range -2^{n-1} through $+2^{n-1}-1$

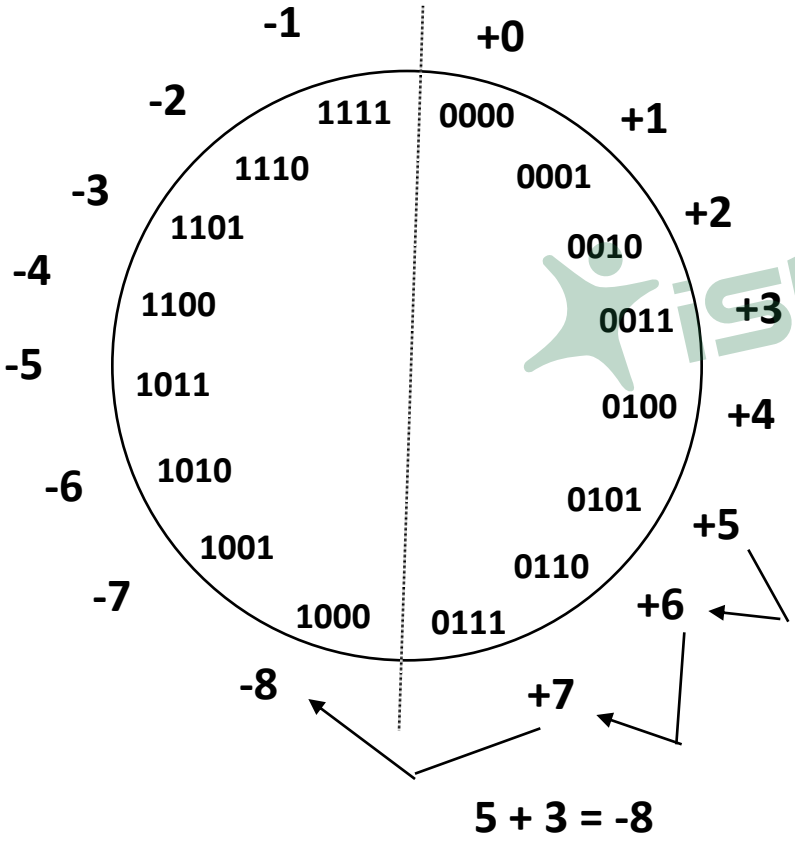
Overflow in Integer Arithmetic

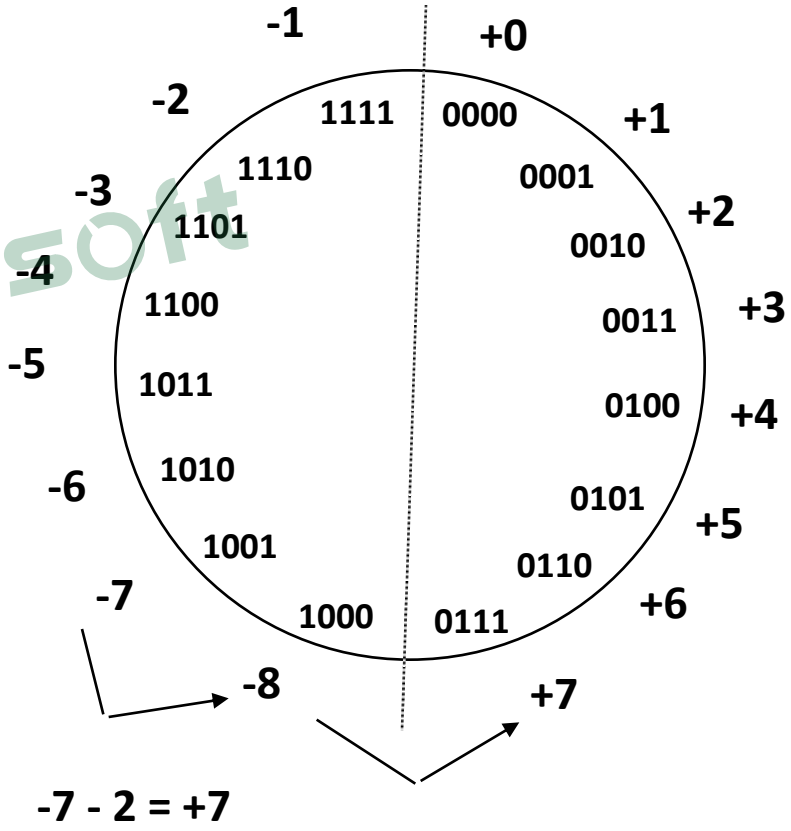
- Overflow can occur only when adding two numbers that have the same sign.

- The carry-out signal from the sign-bit position is not a sufficient indicator of overflow when adding signed numbers.

To detect overflow is to examine the signs of the two summands X and Y and the sign of the result. When both operands X and Y have the same sign, an overflow occurs when the sign of S is not the same as the sign of X and Y.

Overflow - Add two positive numbers to get a negative number or two negative numbers to get a positive number





Overflow Conditions

	0 1 1 1
5	0 1 0 1
3	0 0 1 1
-8	1 0 0 0

Overflow

	1 0 0 0
-7	1 0 0 1
-2	1 1 0 0
7	1 0 1 1 1

Overflow

	0 0 0 0
5	0 1 0 1
2	0 0 1 0
7	0 1 1 1

No overflow

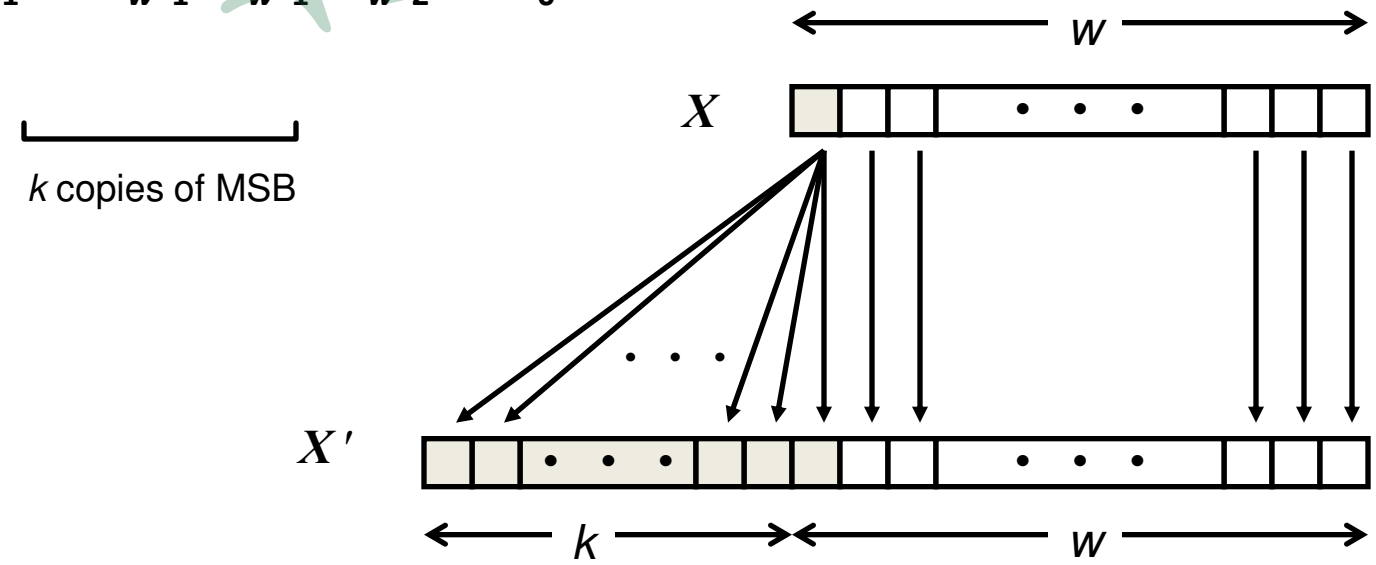
	1 1 1 1
-3	1 1 0 1
-5	1 0 1 1
-8	1 1 0 0 0

No overflow

Overflow when carry-in to the high-order bit does not equal carry out

Sign Extension

- Task:
  - Given *w*-bit signed integer *x*
  - Convert it to *w*+*k*-bit integer with same value
- Rule:
  - Make *k* copies of sign bit:
  - $X' = x_{w-1}, \ldots, x_{w-1}, x_{w-1}, x_{w-2}, \ldots, x_0$

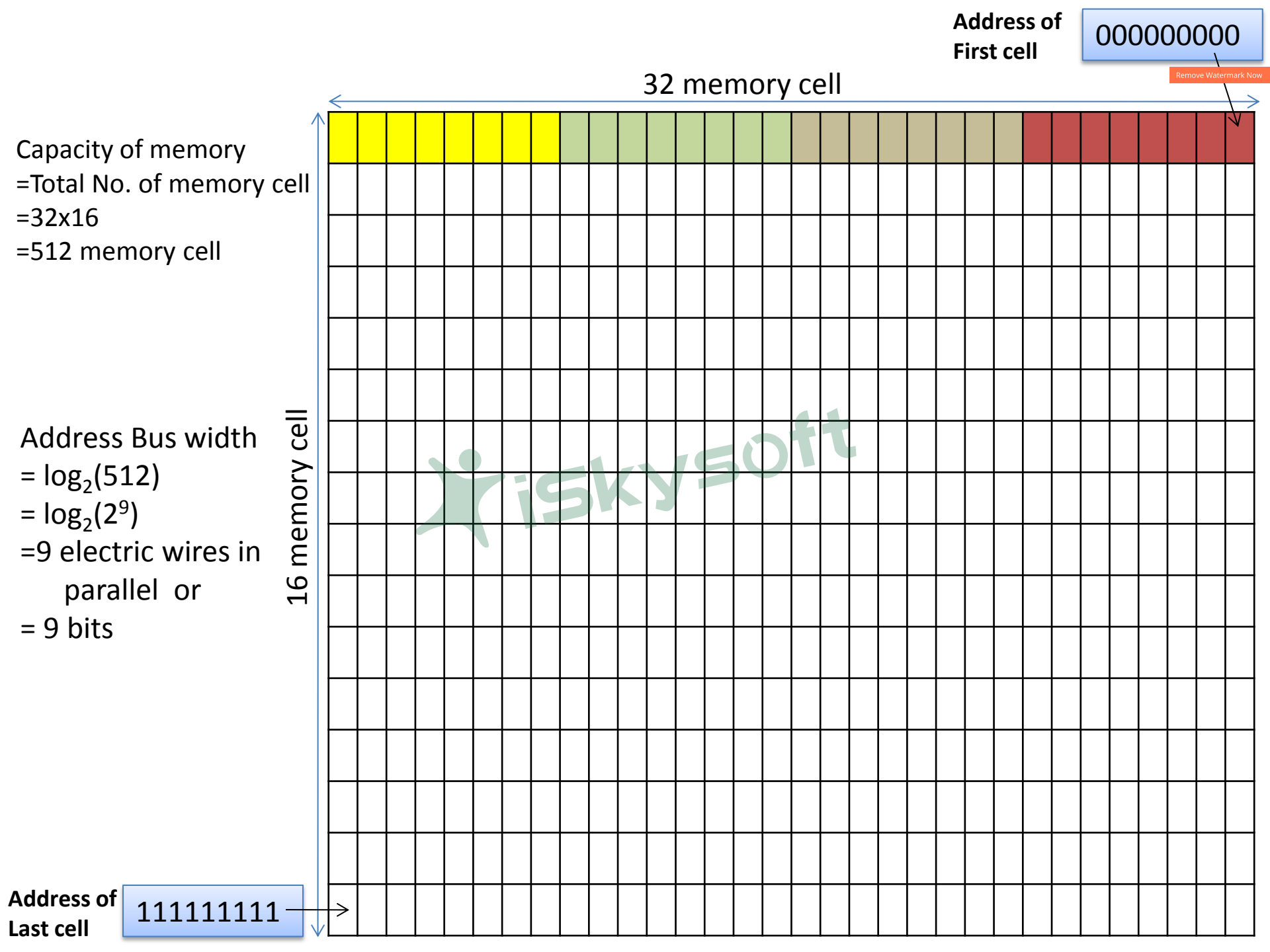


Sign Extension Example

```
short int x =  15213;
int       ix = (int) x;
short int y = -15213;
int       iy = (int) y;
```

	Decimal	Hex	Binary
x	15213	3B 6D	00111011 01101101
ix	15213	00 00 C4 92	00000000 00000000 00111011 01101101
y	-15213	C4 93	11000100 10010011
iy	-15213	FF FF C4 93	11111111 11111111 11000100 10010011

Memory Locations, Addresses, and Operations


Address of
First cell
000000000
Remove Watermark Now
32 memory cell
Capacity of memory
=Total No. of memory cell
=32x16
=512 memory cell
16 memory cell
Address Bus width
= log2(512)
= log2(2^9)
=9 electric wires in
parallel or
= 9 bits
iSkysoft
Address of
Last cell
111111111

32-bit processor

Capacity of memory
=Total No. of groups
=4x16
=64 groups

Address Bus width
= $\log_2(64)$
= $\log_2(2^6)$
=6 electric wires in parallel or
= 6 bits

Data Bus width
= 32 bits

000011 | 000010 | 000001 | Address of First byte or group: 000000

4 groups

16

3	2	1	0
7	6	5	4
11	10	9	8
15	14	13	12
19	18	17	16
23	22	21	20
27	26	25	24
31	30	29	28
35	34	33	32
39	38	37	36
43	42	41	40
47	46	45	44
51	50	49	50
55	54	53	52
59	58	57	56
63	62	61	60

Address of last byte or group: 111111

32-bit processor

4 groups

W_1	0	3	2	1	0
W_2	4	7	6	5	4
W_3	8	11	10	9	8
W_4	12	15	14	13	12
W_5	16	19	18	17	16
W_6	20	23	22	21	20
W_7	24	27	26	25	24
W_8	28	31	30	29	28
W_9	32	35	34	33	32
W_{10}	36	39	38	37	36
W_{11}	40	43	42	41	40
W_{12}	44	47	46	45	44
W_{13}	50	51	50	49	50
W_{14}	52	55	54	53	52
W_{15}	56	59	58	57	56
W_{16}	60	63	62	61	60

32-bit processor

4 groups

W_1	0	2^6-61	2^6-62	2^6-63	2^6- 64
W_2	4	2^6-57	2^6-58	2^6-59	2^6-60
W_3	8	2^6-53	2^6-54	2^6-55	2^6-56
W_4	12	2^6-49	2^6-50	2^6-51	2^6-52
W_5	16	2^6-47	2^6-48	2^6-49	2^6-48
W_6	20	2^6-43	2^6-44	2^6-45	2^6-44
W_7	24	2^6-39	2^6-40	2^6-41	2^6-40
W_8	28	2^6-35	2^6-36	2^6-37	2^6-36
W_9	32	2^6-31	2^6-32	2^6-33	2^6-32
W_{10}	36	2^6-27	2^6-28	2^6-29	2^6-28
W_{11}	40	2^6-23	2^6-24	2^6-25	2^6-24
W_{12}	44	2^6-19	2^6-20	2^6-21	2^6-20
W_{13}	50	2^6-13	2^6-14	2^6-15	2^6-16
W_{14}	52	2^6-9	2^6-10	2^6-11	2^6-12
W_{15}	56	2^6-5	2^6-6	2^6-7	2^6-8
W_{16}	60	2^6-1	2^6-2	2^6-3	2^6-4

32-bit processor

4 groups

W_1	2^6- 64	2^6-61	2^6-62	2^6-63	2^6- 64
W_2	2^6-60	2^6-57	2^6-58	2^6-59	2^6-60
W_3	2^6-56	2^6-53	2^6-54	2^6-55	2^6-56
W_4	2^6-52	2^6-49	2^6-50	2^6-51	2^6-52
W_5	2^6-48	2^6-47	2^6-48	2^6-49	2^6-48
W_6	2^6-44	2^6-43	2^6-44	2^6-45	2^6-44
W_7	2^6-40	2^6-39	2^6-40	2^6-41	2^6-40
W_8	2^6-36	2^6-35	2^6-36	2^6-37	2^6-36
W_9	2^6-32	2^6-31	2^6-32	2^6-33	2^6-32
W_{10}	2^6-28	2^6-27	2^6-28	2^6-29	2^6-28
W_{11}	2^6-24	2^6-23	2^6-24	2^6-25	2^6-24
W_{12}	2^6-20	2^6-19	2^6-20	2^6-21	2^6-20
W_{13}	2^6-16	2^6-13	2^6-14	2^6-15	2^6-16
W_{14}	2^6-12	2^6-9	2^6-10	2^6-11	2^6-12
W_{15}	2^6-8	2^6-5	2^6-6	2^6-7	2^6-8
W_{16}	2^6-4	2^6-1	2^6-2	2^6-3	2^6-4

32-bit processor

4 groups

0	3	2	1	0
4	7	6	5	4
2^k-4	2^k-1	2^k-2	2^k-3	2^k-4

Memory Location, Addresses, and Operation

- Memory consists of many millions of storage cells, each of which can store 1 bit.
- Data is usually accessed in *n*-bit groups. *n* is called word length.

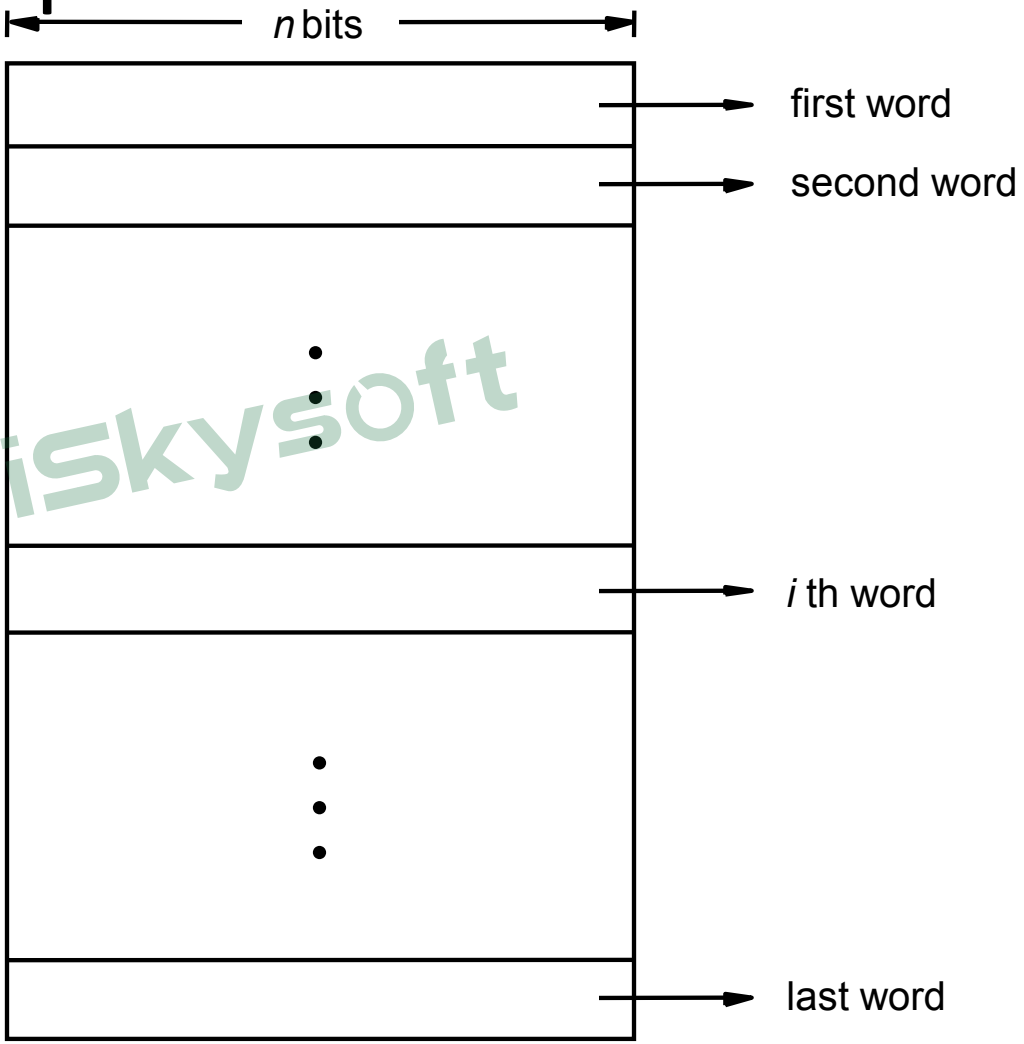
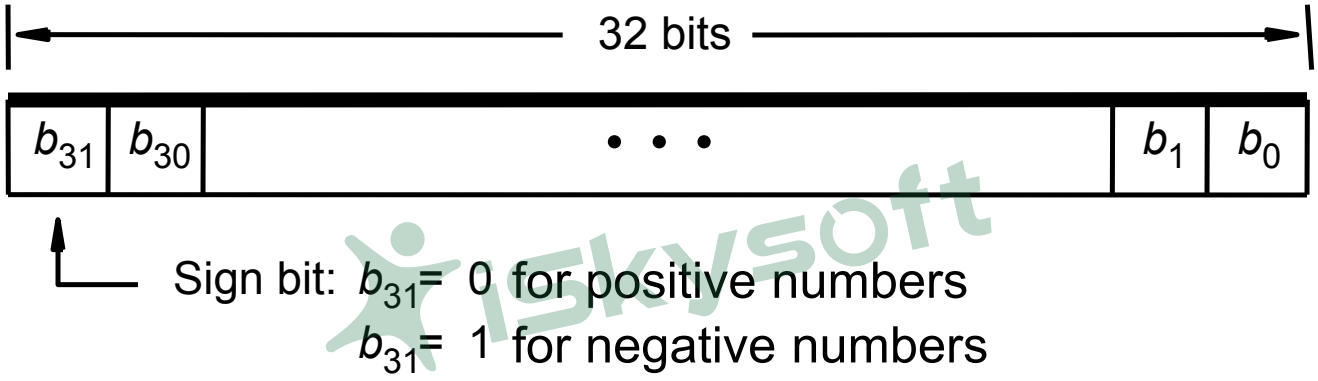


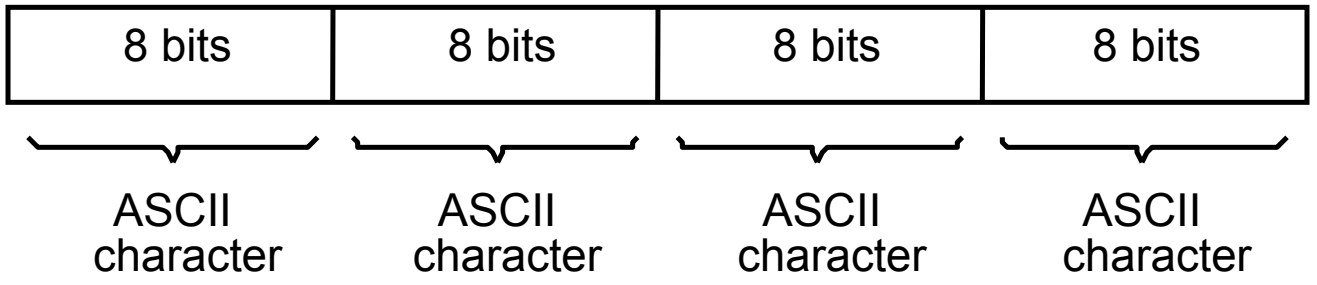
Figure 2.5. Memory words.

Memory Location, Addresses, and Operation

- 32-bit word length example



(a) A signed integer



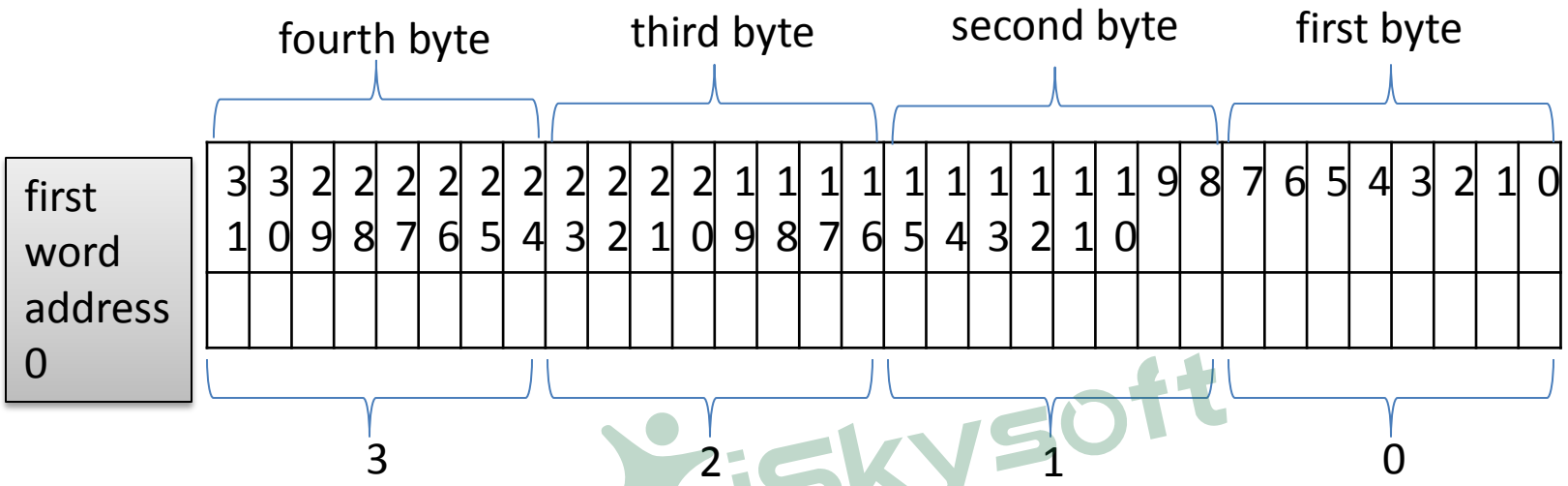
(b) Four characters

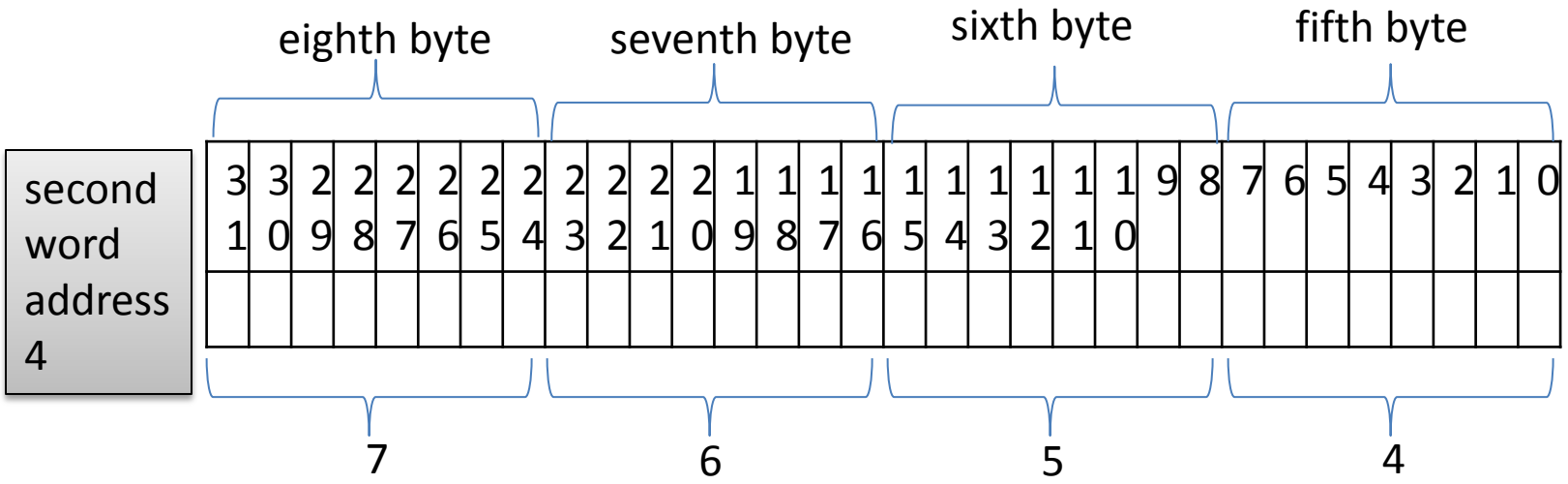
Memory Location, Addresses, and Operation

- To retrieve information from memory, either for one word or one byte (8-bit), addresses for each location are needed.
- A *k*-bit address memory has 2^k memory locations, namely $0 – 2^k-1$, called memory space.
- 24-bit memory: 2^{24} = 16,777,216 = 16M ($1M=2^{20}$)
- 32-bit memory: 2^{32} = 4G ($1G=2^{30}$)
- 1K(kilo)=2^{10}
- 1T(tera)=2^{40}

Memory Location, Addresses, and Operation

- It is impractical to assign distinct addresses to individual bit locations in the memory.
- The most practical assignment is to have successive addresses refer to successive byte locations in the memory – byte-addressable memory.
- Byte locations have addresses 0, 1, 2, … If word length is 32 bits, they successive words are located at addresses 0, 4, 8,…

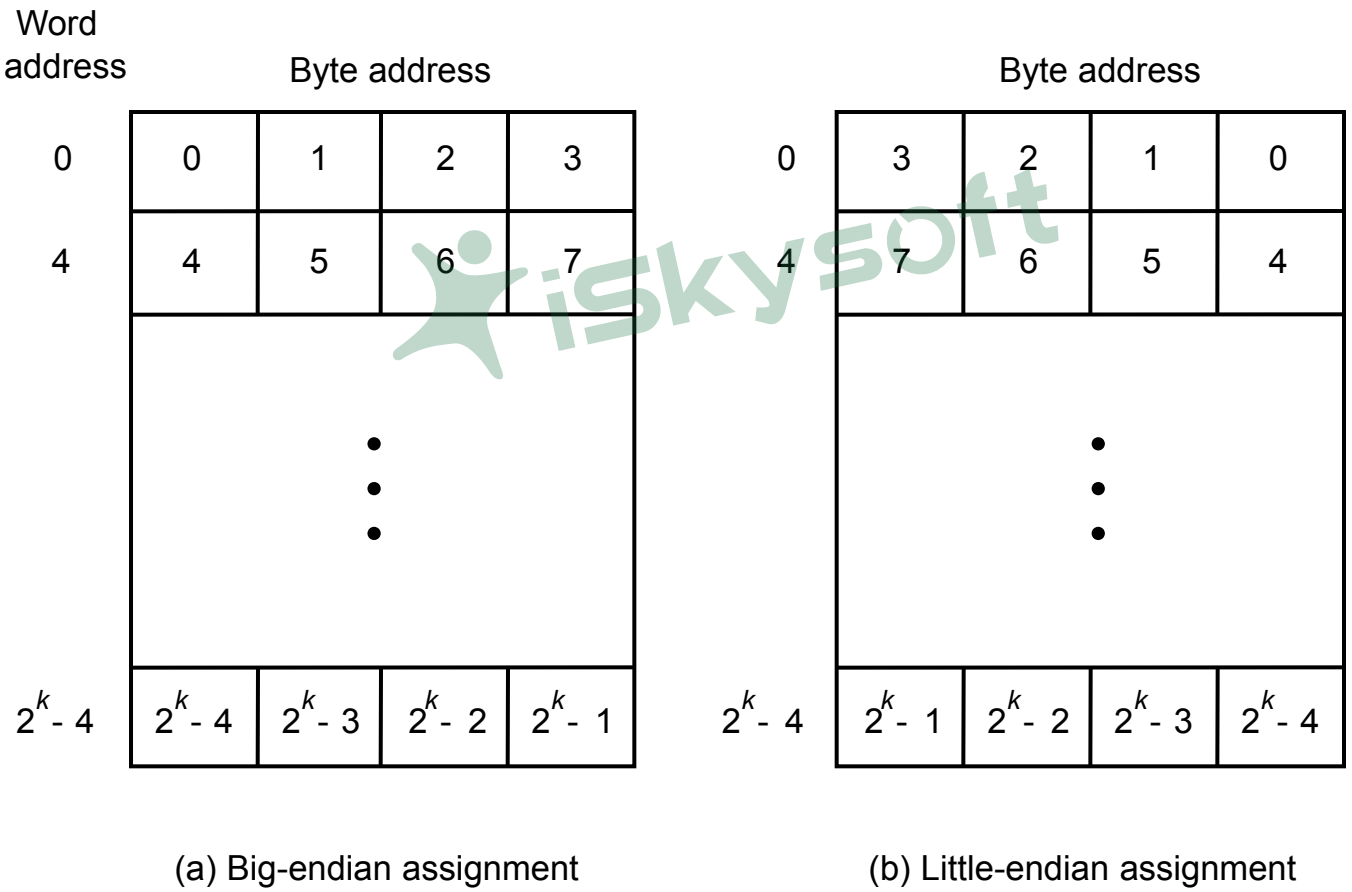




Big-Endian and Little-Endian Assignments

Big-Endian: lower byte addresses are used for the most significant bytes of the word

Little-Endian: opposite ordering. lower byte addresses are used for the less significant bytes of the word

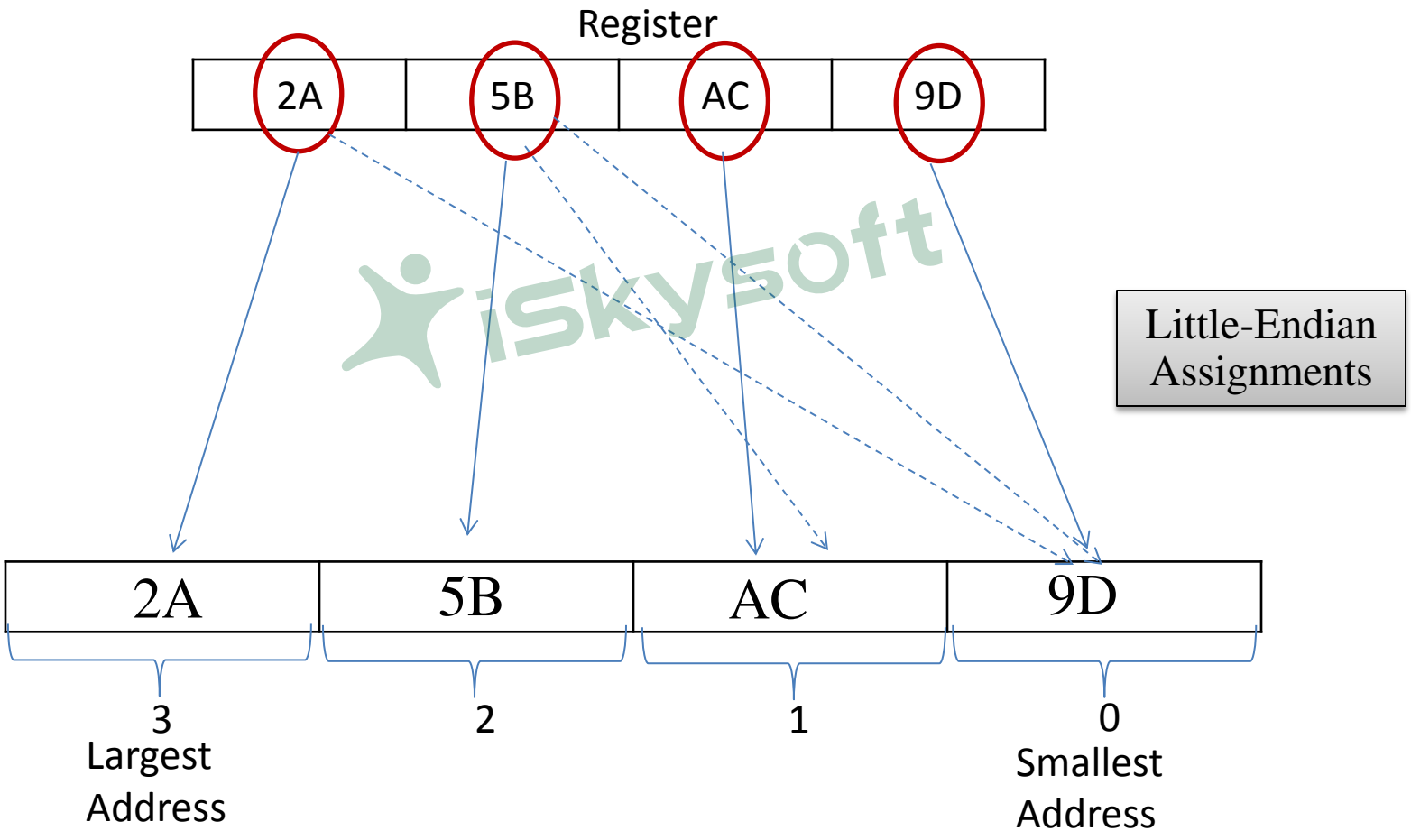


(a) Big-endian assignment

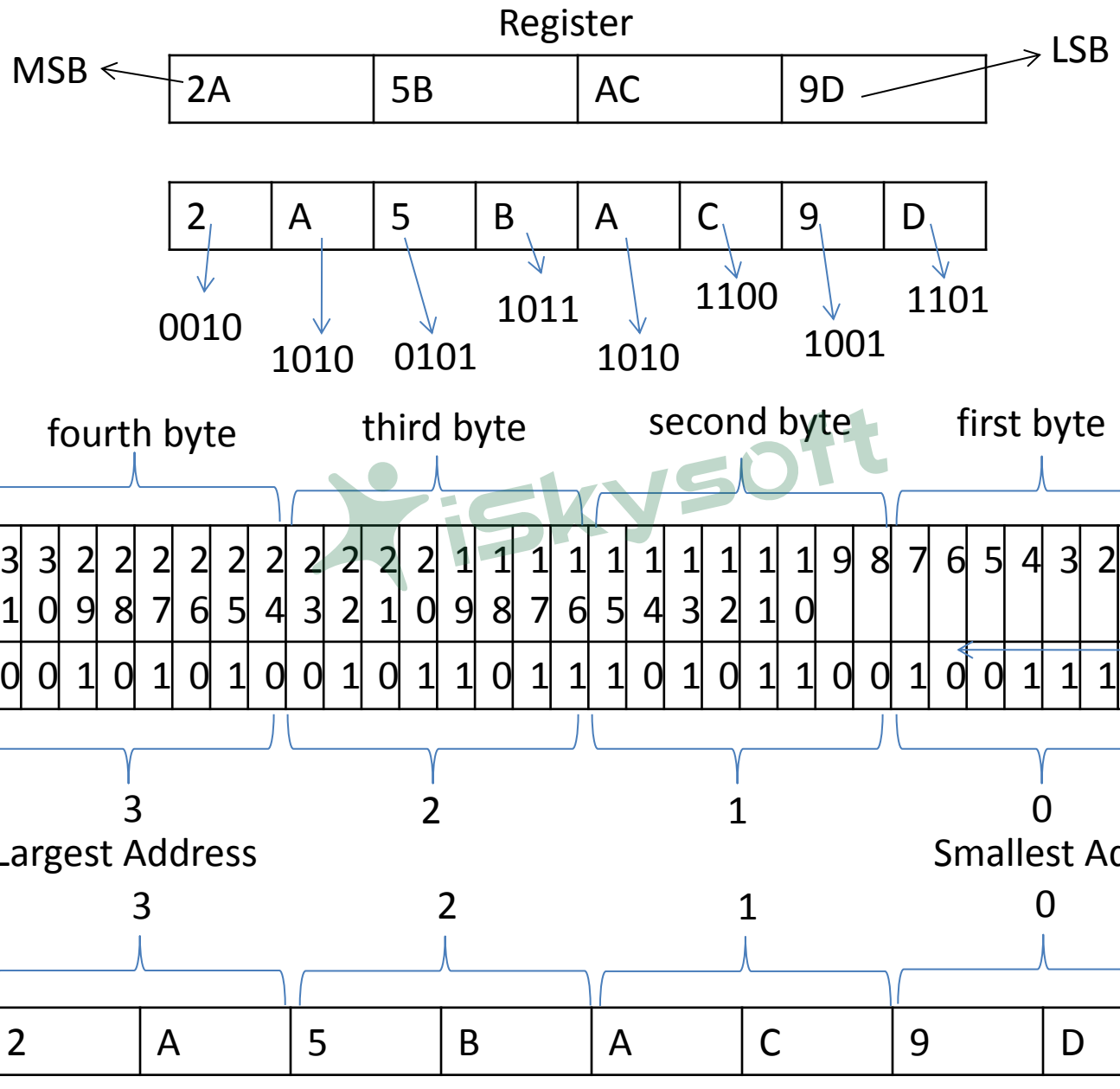
(b) Little-endian assignment

Figure 2.7. Byte and word addressing.

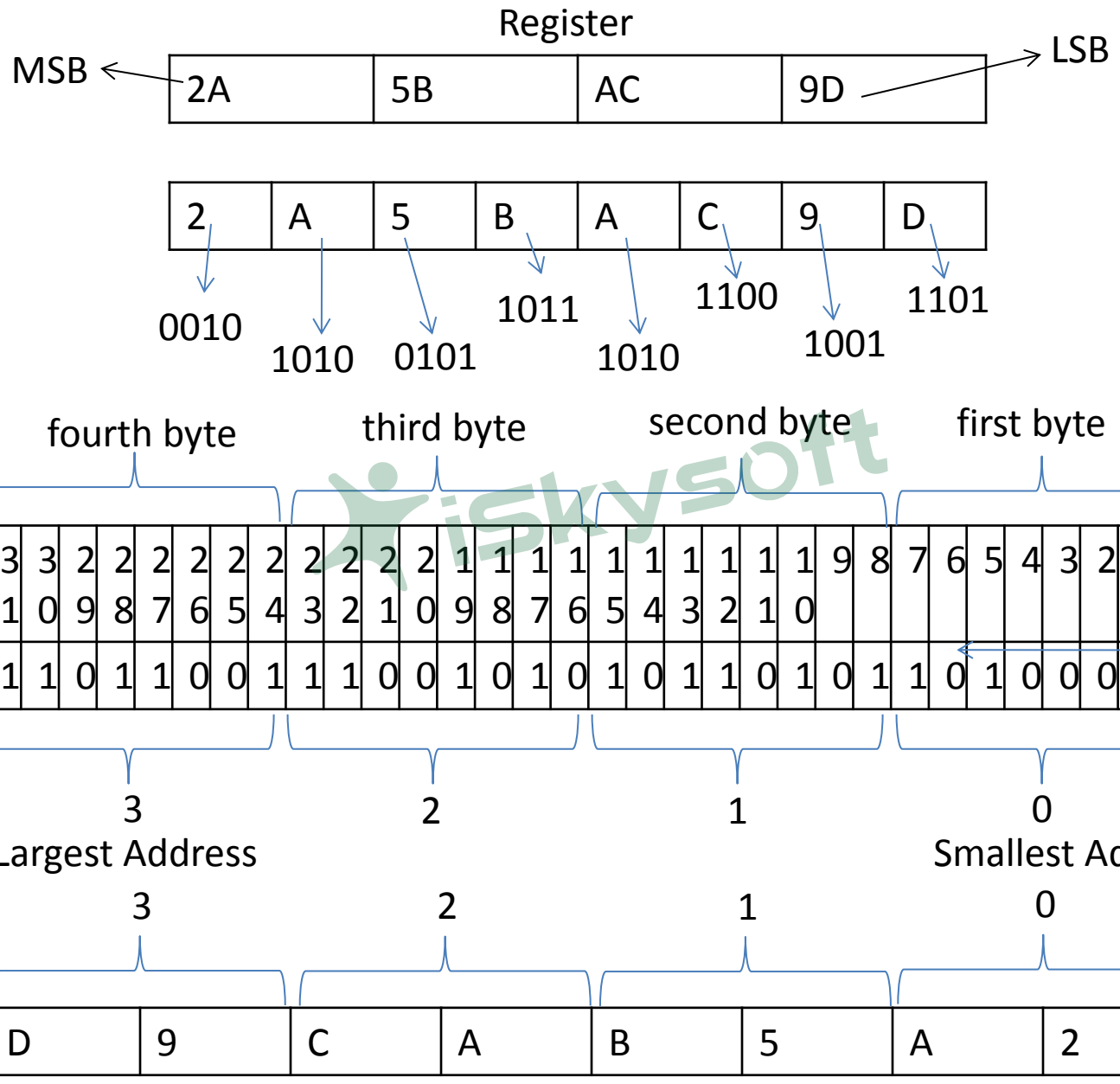
Ordering of bytes in a memory location



Little-Endian Assignments



Big-Endian Assignments



Data Alignment

- Data stored in memory must be “aligned” according to the length of the data
  - Byte: 8 bits
    - can go at any address
  - Word: 16 bits
    - must be “word aligned”
    - stored at even number addresses
  - Long word: 32 bits
    - must be “long word” aligned
    - stored at addresses (i.e. divisible by 4)
  - Double word: 64 bits
    - must be “double word aligned”
    - addresses must be divisible by 8

word addresses end in a number divisible by 2: 0, 2, 4, 6, 8, A, C,.. etc..

00000003	00000002	00000001	00000000
00000007	00000006	00000005	00000004
0000000B	0000000A	00000009	00000008
0000000F	0000000E	0000000D	0000000C
00000013	00000012	00000011	00000010
00000017	00000016	00000015	00000014
0000001B	0000001A	00000019	00000018

- Address ordering of bytes
- Word alignment
  - Words are said to be aligned in memory if they begin at a byte addr. that is a multiple of the num of bytes in a word.
    - 16-bit word: word addresses: 0, 2, 4,….
    - 32-bit word: word addresses: 0, 4, 8,….
    - 64-bit word: word addresses: 0, 8,16,….
- Access numbers, characters, and character strings

- Functions and characteristic of Instructions

- Categories of instructions:-
  1. Data move type instructions
  2. Arithmetic and logical instructions
  3. Program flow control and machine control instructions

Memory Operation

- Load (or Read or Fetch)
  - Copy the content. The memory content doesn't change.
  - Address – Load
  - Registers can be used
- Store (or Write)
  - Overwrite the content in memory
  - Address and Data – Store
  - Registers can be used

Instruction and Instruction Sequencing

“Must-Perform” Operations

- Data transfers between the memory and the processor registers
- Arithmetic and logic operations on data
- Program sequencing and control
- I/O transfers

Register Transfer Notation

- Identify a location by a symbolic name standing for its hardware binary address (LOC, R0,…)
- Contents of a location are denoted by placing square brackets around the name of the location
- R1←[LOC]
- R2←[LOD]
- R3 ←[R1]+[R2]
- Register Transfer Notation (RTN)

Memory Chip

Address	Data
LOC	10
LOD	20

Instruction Formats

- Three-Address Instructions
  - ADD R1, R2, R3 R3 ← R1 + R2
- Two-Address Instructions
  - ADD R1, R2 R2 ← R1 + R2
- One-Address Instructions
  - ADD M AC ← AC + M[AR]
- Zero-Address Instructions
  - ADD TOS ← TOS + (TOS – 1)
- RISC Instructions
  - Lots of registers. Memory is restricted to Load & Store