JANUARY 2012

Chapter-1

January 01 Sunday

Registers Transfer & Microoperations:

Registers:- A register is a group of flip-flops with each flip-flop capable of storing one bit of information. An n-bit register has a group of n flip-flops and is capable of storing any binary information of n bits. The transfer of new information into register is referred to as loading the register. If all the bits of the reg. are loaded simultaneously with a common clock pulse, we say that loading is done in parallel.

January 02 Monday

Micro operation:- The operations performed on the data stored inside the register is called micro-operations.

e.g:- Add, shift, load, clear etc.

The operations performed on the data stored in register is faster than the data stored in memory.

If we want to TIP occur only under control condition. This can be done with (If-then)

If (P=1) then (R2 ← R1)

P is a control signal and P control signal → control function.
→ control fn is a boolean variable that is equal to 1 or 0.

JANUARY 2012

(2)

January 03 Tuesday

Register Transfer language:

The symbolic notation used to describe the micro-opⁿ among registers is called a RTL.

$$R2 := R1$$

RTL → R2 ← R1

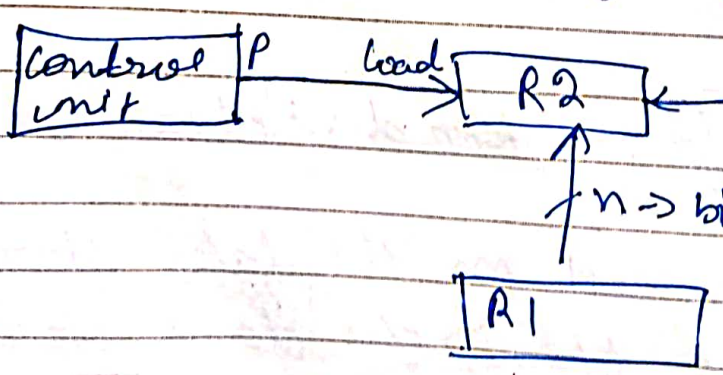
The Reg T/F language method to represent micro-opⁿ with control signal

Control variable generation by control unit

January 04 Wednesday

P: R2 ← R1

That means this micro-opⁿ is executed when control & Timing signal both are 1.



Reg. T/F opⁿ.

MAR → memory address register
PC → program counter
IR → Instruction Register
R1 → processor register.

January 05 Thursday

Basic Symbols for Reg. T/F.

Representation of reg:-

Computers registers are designed by capital letters (sometimes followed by numerals) to denote the function of the register.

e.g:-

→ The reg. that holds an address for the memory unit is designed by the name MAR.

→ The PC reg. that hold the address of the nxt

January 06 Friday

Inst. to be executed.

→ IR that holds the inst. under execution.

→ AC → Result Storage

R1 → Processor Register.

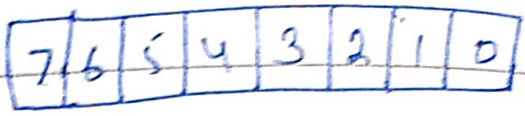
→ The individual flip-flop in an n-bit register are numbered in sequence 0 to n-1 starting from 0 in the rightmost position to the left.

JANUARY 2012

January 07 Saturday

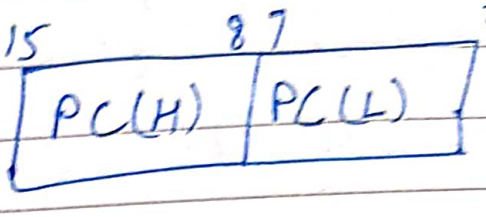
(4)

Register R1



showing individual bits of an 8-bit.

Numbering of bits of 16 bits.



Low byte & high byte.

January 16 Monday

Microoperations:

1. Register T/F Micro-op^n.
2. Arithmetic micro-op^n.
3. Logic micro-op^n.
4. Shift micro-op^n.

Arithmetic Micro-op^n:

1) $R3 \leftarrow R1+R2$ ⇒ Contents of R1 Plus R2 Transferred to R3
2) $R3 \leftarrow R1-R2$

January 17 Tuesday

3) $R2 \leftarrow \overline{R2}$ ⇒ complement the contents of R2 (1's complement)
4) $R2 \leftarrow \overline{R2}+1$ ⇒ 2's complement of R2.
5) $R1 \leftarrow R1+1$ ⇒ Increment the contents of R1 by one
6) $R1 \leftarrow R1-1$ ⇒ Decrement the contents of R1 by one.

These type of micro operations are used to perform the bit manipulation operations on data available in registers.

1100
0110
1110

JANUARY 2012

(6)

January 20 Friday

Logic Micro op^n:-

Different in meaning?

logic micro-op^n specify binary operations for string of bits stored in registers.

E.g.:- R1 & R2 → Two registers.

P: R1 ← R1 ⊕ R2

list of logic Micro-op^n:-

F ← 0	Clear	✓
F ← A ∧ B	AND	(X) ✓
F ← A	Transfer A	✓

January 21 Saturday

F ← B	Transfer B	
F ← A ⊕ B	Exclusive-OR	
F ← A ∨ B	OR	(+) ✓
F ← $\overline{A} \vee B$	NOR	✓
F ← $\overline{A \oplus B}$	Exclusive-NOR	
F ← $\overline{B}$	Complement B	
F ← $\overline{A \wedge B}$	NAND	✓

January 22 Sunday

F ← all 1's	Set to all 1's	✓

January 23 Monday

Selective-Set:

It sets the bits in reg A to 1, where there are 1's in corresponding bit of reg. B. This does not effect the bit of reg. B. This does not effect the bit value in A where there are 0's in corresponding bit of register B. This op^n is equivalent to logical OR.

E.g:

```
1001   Reg A
1010   Reg B
----
1011   Reg A after operation.
```

$A \leftarrow A \vee B$

January 24 Tuesday

2. Selective complement op^n:- It complements bits in A where there are corresponding 1's in B. It does not effect bit positions that have 0's in B.

E.g:

```
1010   Reg A
1100   Reg B
----
0110   A after
```

The Exclusive-OR micro-op^n can be used to selective complement bits of a register

January 25 Wednesday **Selective clear:-** This operation clears to 0 the bits in A only only where there corresponding 1's in B.

e.g:

```
1010    Reg A
1100    Reg B
0010    A Reg. After.
```

Logic micro-operation is

$$A \leftarrow A \wedge \bar{B}$$

January 26 Thursday **Shift Microoperations:-**

Shift microoperations are used for serial transfer of data. The contents of a register can be shifted to the left or the right.

There are three types of shift. i'e

1. Logical
2. circular
3. Arithmetic

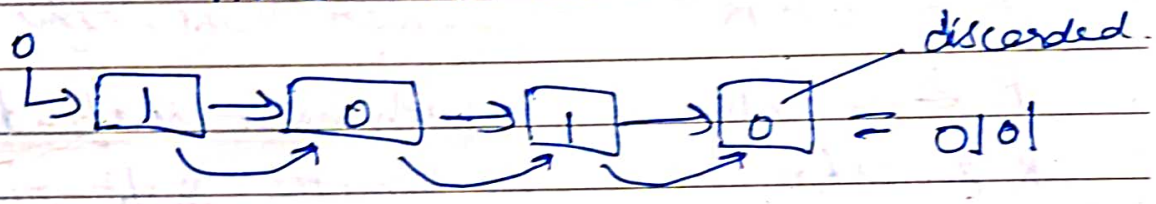
January 27 Friday ① Logical Shift: A logical shift is one that transfer the serial input. The symbols used for shift. A logical shift is that T/F 0 thro. the serial input

Shl → logical shift left

Shr → logical shift right.

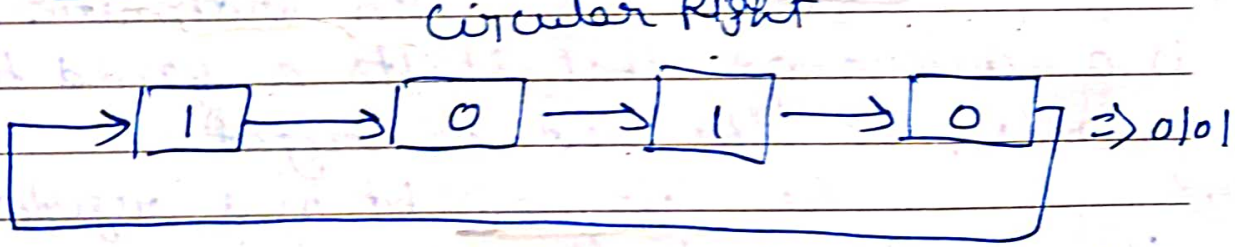
Eg: R1 ← Shl R1

R2 ← Shr R2.



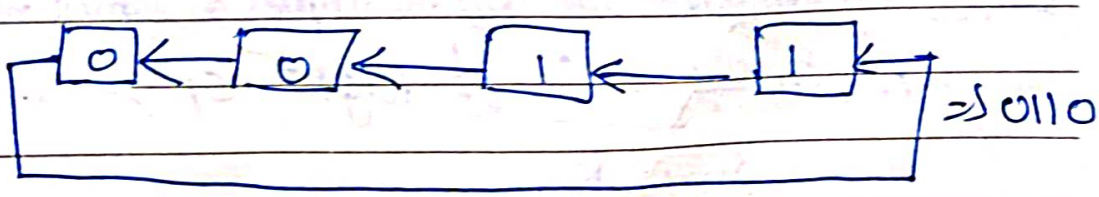
January 28 Saturday

Circular Shift:



R1 ← Cir R1

January 29 Sunday



Circular left

R1 ← Cil R1

January 30 Monday The circular shift (rotate operation) circulates the bits of the register around the two ends without loss of information.

List of Shift micro-op's:-

R ← shl R	Shift-left register R
R ← shr R	Shift-right "
R ← cil R	Circular Shift-left Reg. R.
R ← cir R	Circular Shift-right "
R ← ashl R	Arithmetic shift-left R.
R ← ashr R	Arithmetic shift-right R.

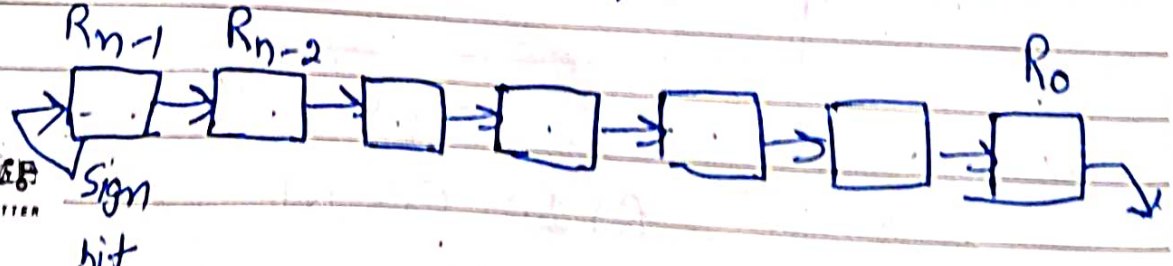
January 31 Tuesday

Arithmetic Shift :-

An arithmetic shift is a micro-op^n that shifts a signed binary number to the left or right.

The left-most bit in a register holds the sign bit and the remaining bits hold the number. The sign bit is 0 for +ve and 1 for -ve. An arithmetic left shift multiplies a signed no. by 2 and an

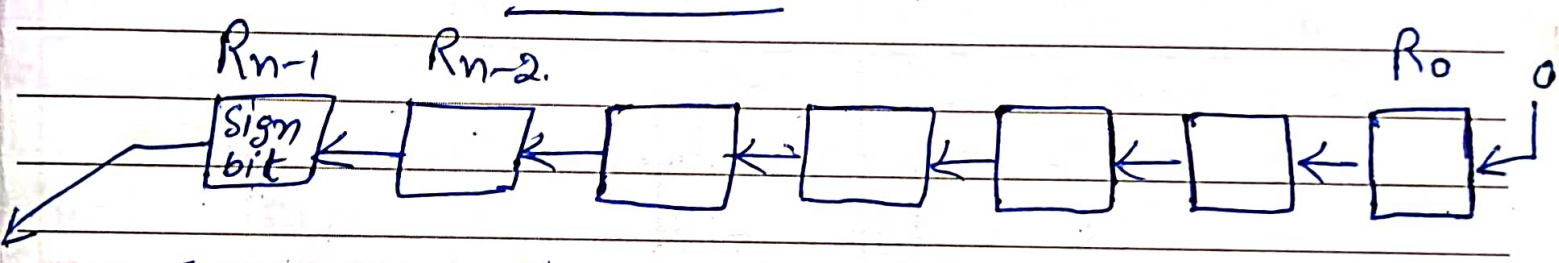
Arithmetic Shift Right op^n.



Arithmetic right shift divides a signed no. by 2

Notes The arithmetic shift-right leaves the sign-bit unchanged and shifts the number (including the sign bit) to the right. If a Register R has n bits (denoted from 0 to n-1 from right to left), then the bit R_{n-1} remains unchanged. R_{n-2} receives the bit from R_{n-1} and so on for the other bits in the register. The bit R_0 is lost.

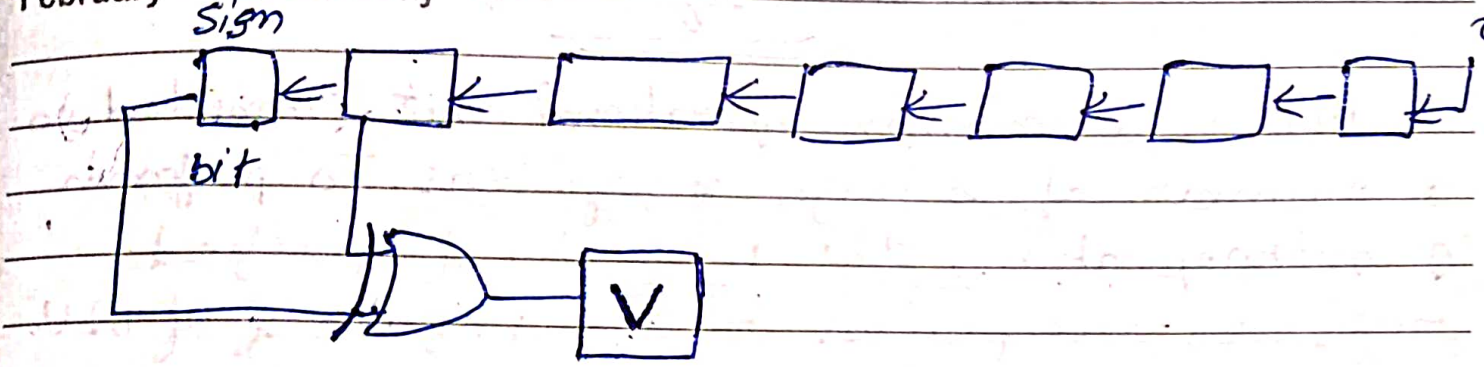
Arithmetic Shift-Left operation :-



The arithmetic shift-left operation inserts a 0 into R_0 and shifts all other bits to the left. The initial bit of R_{n-1} is lost and replaced by the bit from R_{n-2}. ~~A sign reversal occurs if the~~ An overflow occurs after an arithmetic shift left if initially, before the shift, R_{n-1} is not equal to R_{n-2}. An overflow flip-flop V can be used to detect an arithmetic shift-left overflow.

XOR 0 0 → false, 0 1 → true, 1 0 → true, 1 1 → false

February 01 Wednesday



Detecting overflow in arithmetic shift-left operation.

$$V = R_{n-1} \oplus R_{n-2}$$

XOR gate

0 1 false
0 1 false
1 0 true

If V=0, there is no overflow, but if V=1, there

February 02 Thursday is an overflow which indicates a sign reversal after the shift.

Right shift

+10

e.g. 01010 — 10 ie 10 got divided by 2.

00101 — 5

Left Shift

+10

01010 — 10

10100 — 20 ie 10 got multiplied by 2.

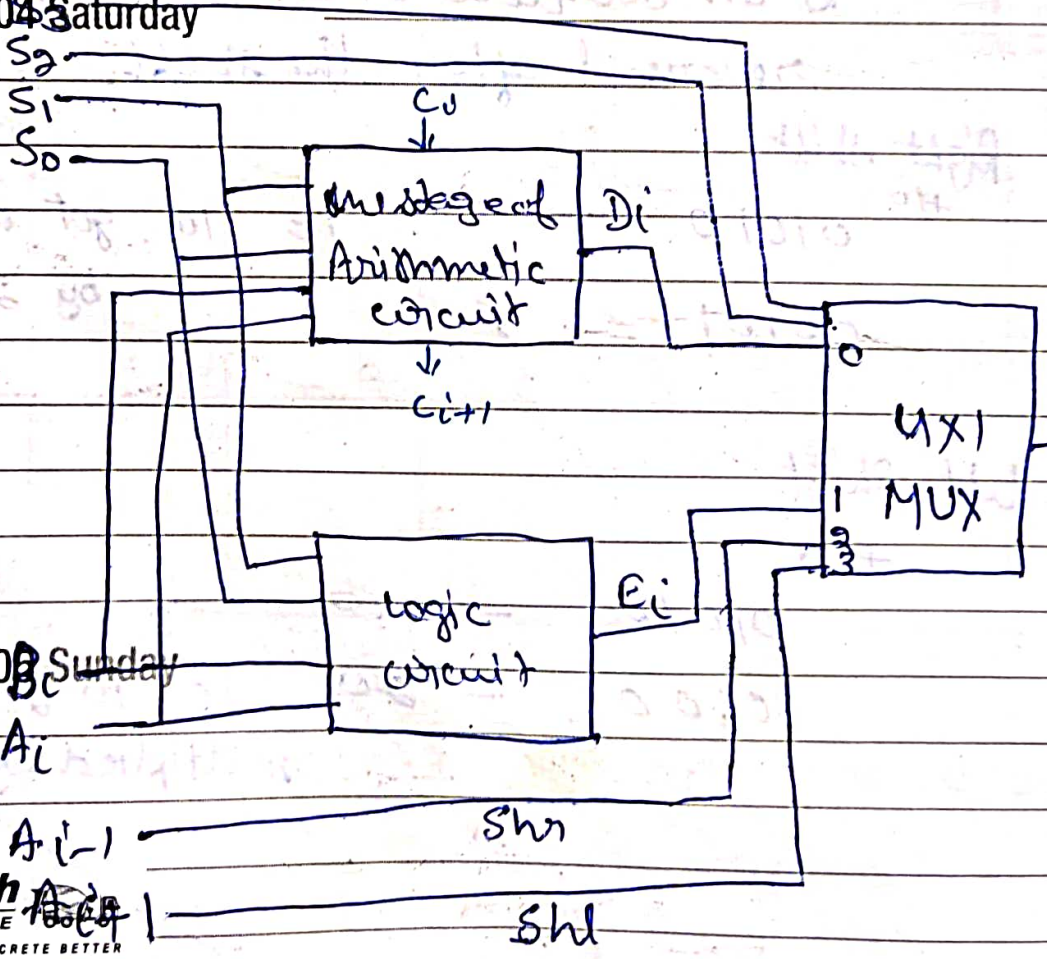
February 03 Friday

Arithmetic Logic Shift Unit :

ALU is a common operational unit connected to a number of storage registers. To perform a microoperation, the contents of specified registers are placed in the inputs of ALU. The ALU performs an operation and the result is then transferred to a destination registers.

The arithmetic logic shift unit consist of arithmetic circuit, the logic circuit and the shift circuit. The circuits are combined with common selection variables.

February 04 Saturday



February 05 Sunday

February 08 Wednesday

S3	S2	S1	S0	Cin	Operation	Function
0	0	0	0	0	F = A	T/F A
0	0	0	0	1	F = A+1	Increment A
0	0	0	1	0	F = A+B	Addition
0	0	0	1	1	F = A+B+1	Add with carry
0	0	1	0	0	F = A+B'	Sub. with borrow
0	0	1	0	1	F = A+B'+1	Subtraction
0	0	1	1	0	F = A-1	Decrement A
0	0	1	1	1	F = A	T/F A
0	1	0	0	X	F = A ∧ B	AND
0	1	0	1	X	F = A ∨ B	OR
0	1	1	0	X	F = A ⊕ B	XOR
0	1	1	1	X	F = Ā	complement A

February 09 Thursday The inputs A_i and B_i are applied to both the arithmetic and logic units. A particular micro-operation is selected with inputs S_1 & S_0. A 4x1 multiplexer at the output chooses b/w arithmetic output in D_i & logic output in E_i. The data in multiplexer are selected with inputs S_3 & S_2. The other two data inputs to the multiplexer receive inputs A_{i-1} for shift-right and A_{i+1} for shift-left. The output carry C_{i+1} of a arithmetic stage must be connected to the input carry C_i of the next stage in sequence. The circuit shows 8 arithmetic operation, four logic operation and two shift operation

Arithmetic operations are selected with $S_3 S_2 = 00$

Logic operations are selected with $S_3 S_2 = 01$

Shift operations are selected with $S_3 S_2 = 10$ & 11

1	0	X	X	F = Shr A	Shift right A into F
1	1	X	X	F = Shl A	Shift left A into F.