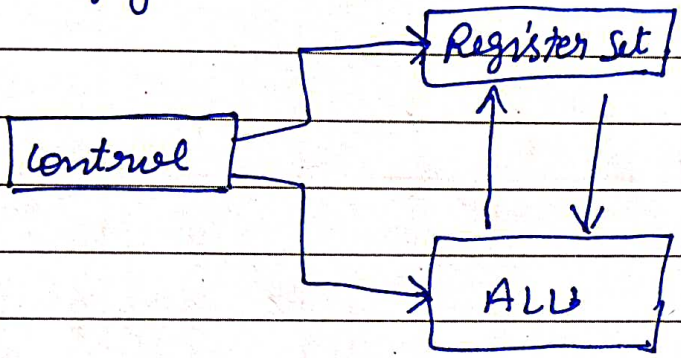
Notes

Central Processing Unit (CPU)

Introduction:

CPU: The part of the computer that performs the bulk of data-processing operations is called the CPU. The CPU is made up of three major parts as shown in fig:

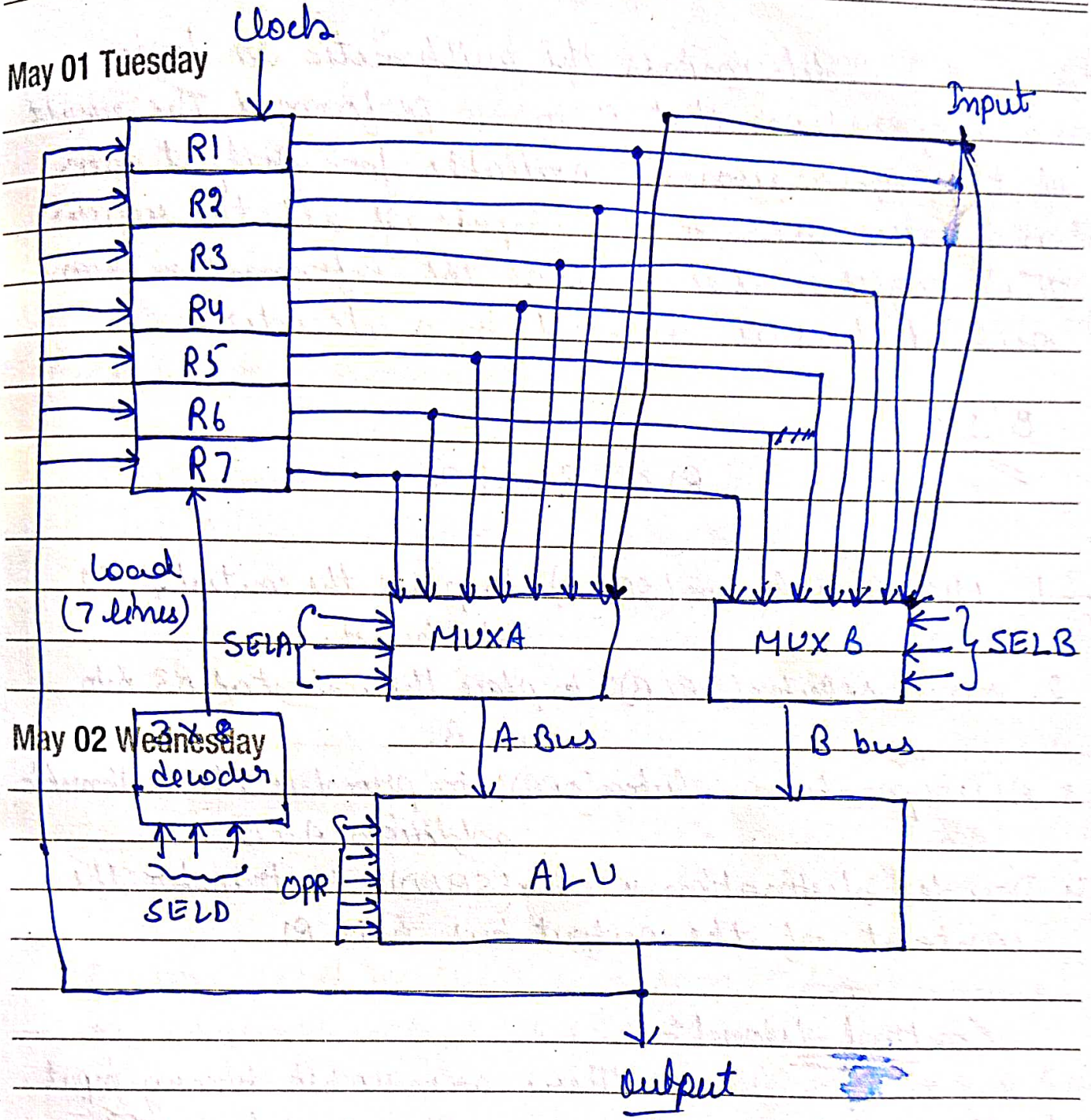


* The Register set stores intermediate data used during the execution of the instructions. The ALU performs the required micro-operations for executing the instructions. The control unit supervises the transfer of information among the registers and instructs the ALU as to which operation to perform.

General Register Organization:

A bus organization for seven CPU registers as shown in fig. The output of each register is connected to two multiplexers to form the two buses A and B. The selection lines in each multiplexers select one register or the input data for particular bus.

May 01 Tuesday



May 02 Wednesday

3	3	3	5
SELA	SELB	SELD	OPR

control word

The A and B buses form the inputs to a ALU. The operation selected in the ALU

May 03 Thursday determines the arithmetic or logic micro-operation that is to be performed. The result of the operation is available for output data and also goes to the inputs of all the registers. The register that receives the information from output bus is selected by a decoder.

e.g:-

$R1 \leftarrow R2 + R3$ . . .

1. MUX A selector (SELA): to place the content of R2 into bus A.
2. MUX B selector (SELB): to place the content of R3 into bus B.

May 04 Friday

3. ALU operation selector (OPR): to provide the arithmetic addition A + B.
4. Decoder destination selector (SELD): to transfer the content of the output bus into R1.

Control Word:-

There are 14 bit binary input in the unit and these combined value specifies a control word. It consist of four fields. Three fields contain three bits each and one field has five bits.

↓ Three bits of SELA select a source register for A input of the ALU

May 05 Saturday * The Three bits of SELB select a register for B input of the ALU. The three bits of SELD select a destination register using decoder.

* The five bits of OPR select one of the operations in the ALU. The 14-bit control word when applied to the selection inputs specify a particular microoperation.

Binary code	SELA	SELB	SELD
000	Input	Input	None
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

May 06 Sunday

* When SELA or SELB is 000, the corresponding multiplexer select the external input data.
* When SELD = 000, no destination register is selected.

ALU operations:

OPR	operation	Symbol
00000	TIF A	TSFA
00001	Increment A	INCA
00010	Add A+B	ADD
00101	Subtract A-B	SUB

MAY 2012

May 07 Monday

00110	Decrement	DECA
01000	AND A, B	AND
01010	OR A, B	OR
01100	XOR A, B	XOR
01110	complement	COMA
10000	Shift right A	SHRA
11000	Shift left A	SHLA

Example:

R1 ← R2 − R3

Field:	SELA	SELB	SELD	OPR
Symbol:	R2	R3	R1	SUB
Control Word:	010	011	001	00101

May 08 Tuesday

1) R1 ← R2 − R3
2) R4 ← R4 ∨ R5
3) R6 ← R6 + 1
4) R7 ← R1
5) R4 ← Shl R4

	110	000	110	00001
	001	000	111	00000
	100	000	100	11000

MAY 2012

May 09 Wednesday **Stack Organization:-** A useful feature that is included in the CPU is a stack or last-in, first out (LIFO). A stack is a storage device that stores information in such a manner that the item stored last is the first item retrieved.

The register that holds the address for the stack is called stack pointer (SP) because its value always points at the top item in the stack.

The two operations of a stack are the insertion and deletion of items. The operation of insertion is called push. The operation

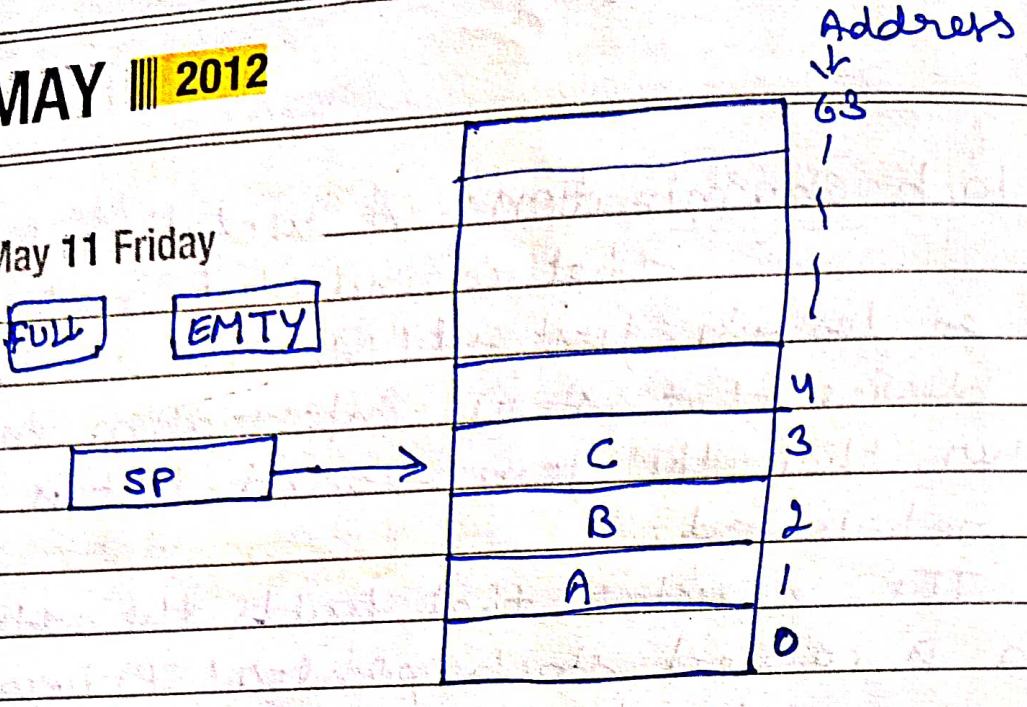
May 10 Thursday of deletion is called pop.

Register Stack:-

A stack can be placed in a portion of a memory or it can be organized as a collection of no. of registers. The SP contains a binary number whose value is equal to the address of the word that is currently on the top of the stack. Three items are placed in the stack: A, B, and C. Item C is on top of the stack so that the content of SP is now 3. To remove the top item, the stack is popped by reading the word at address 3 and decrementing the content of SP.

MAY 2012

May 11 Friday



6 bit address

64-word Stack

In a 64-word stack, the SP contains 6 bits because $2^6 = 64$. Since SP has only six bits, it

May 12 Saturday cannot exceed a no. greater than 63. When 63 is incremented by 1, the result is 0 since $111111 + 1 = 1000000$ in binary, but SP can consider only six least significant bits. The 1 bit register FULL is set to 1 when the stack is full, and the one-bit register EMTY is set to 1 when the stack is empty. DR is a data register that holds the data to be written into or read out of the stack.

Initially, SP is cleared to 0, EMTY is

May 13 Sunday set to 1, and FULL is set to 0, so that SP points to the word at address 0 and stack is marked empty. If the stack is not full, a new item can be inserted with a push operation. The push operation is implemented as follow:

MAY 2012

May 14 Monday

SP ← SP+1 Increment SP

M[SP] ← DR Write item on top of the stack.

If (SP=0) then (FULL ← 1) Check if stack is full.

EMTY ← 0 Mark the stack not empty.

The first item stored in stack at address 1. The last item is stored at address 0. If SP reaches 0, the stack is full of items, so FULL = 1.

A new item is deleted from the stack if the stack is not empty.

May 15 Tuesday

DR ← M[SP] Read item from the top of stack.

SP ← SP-1 Decrement SP

If (SP=0) then (EMTY ← 1) check if stack is empty.

FULL ← 0

Memory Stacks: Memory stacks can be implemented in a RAM attached to CPU. The fig. shows a portion of memory partitioned into three segments ie program, data and stack.

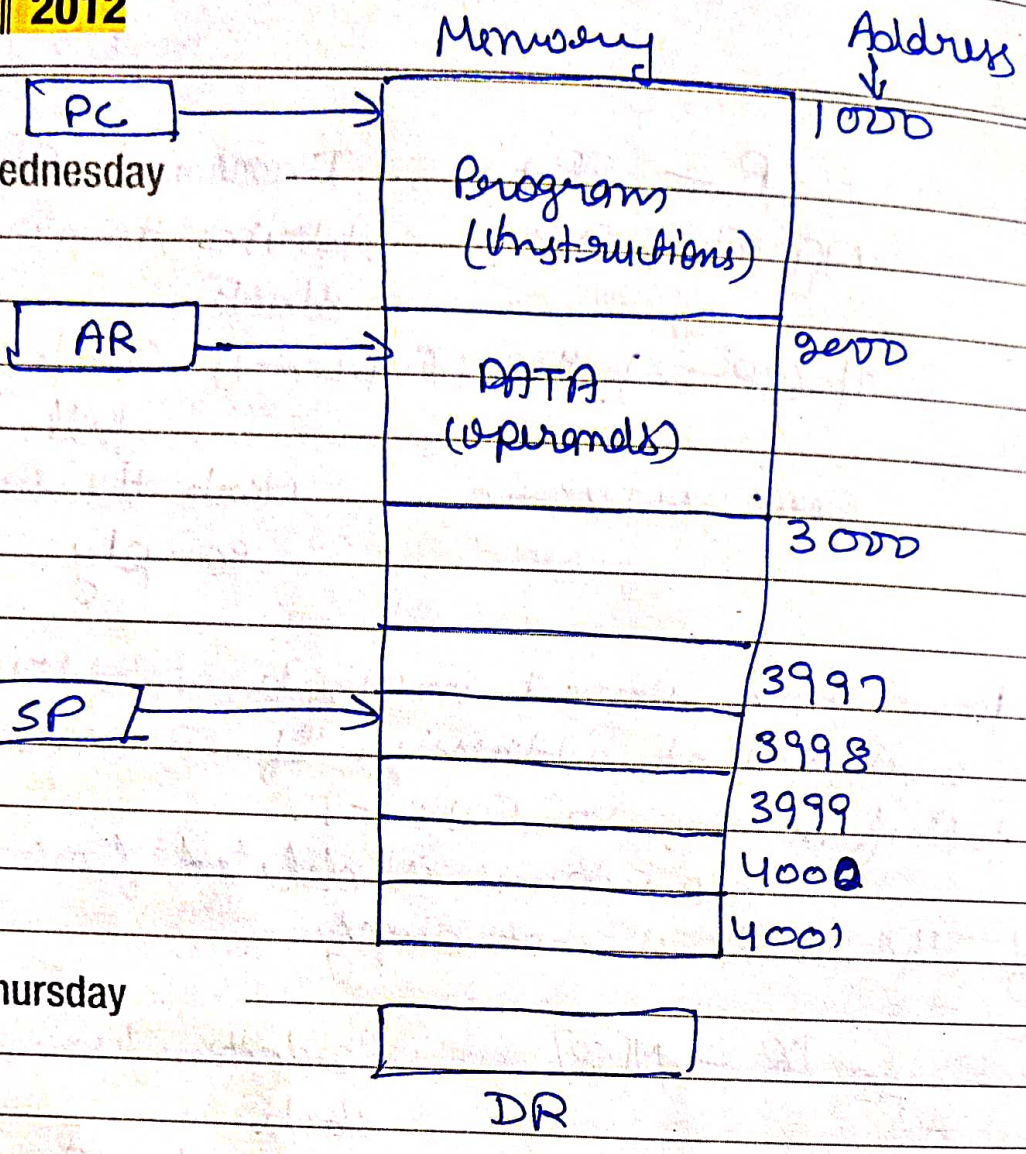
⇒ PC for address of next inst.

⇒ AR for data

⇒ SP for top of the stack.

MAY 2012

May 16 Wednesday



May 17 Thursday

A new item is inserted with Push operation:

$SP \leftarrow SP - 1$

$M[SP] \leftarrow DR$

A new item is deleted with a pop operation:

$DR \leftarrow M[SP]$

$SP \leftarrow SP + 1$

MAY 2012

Stack overflow or underflow:

May 18 Friday The stack ~~overflow~~ limits can be checked by using two processor registers: One to hold the upper limit (3000) and other to hold the lower limit (4001). After a push operation, SP is compared with the upper limit register and after a pop operation, SP is compared with the lower limit register.

Instruction format (Address field format)

1. Three-Address Instruction
2. Two " "
3. One - " "

May 19 Saturday 4. zero Address Inst.

$$X = (A+B) * (C+D)$$

1. Three-Address Instruction:

Computer with three-address instruction formats can use each address field to specify either a processor register or a memory operand.

e.g:

May 20 Sunday

```
ADD  R1, A, B   R1 ← M[A] + M[B]
ADD  R2, C, D   R2 ← M[C] + M[D]
MUL  X, R1, R2  M[X] ← R1 * R2
```

May 21 Monday * Advantage of three 3-Address format is that it results in ~~it results in~~ short programs when evaluating arithmetic expressions.

* The disadvantage is that the binary coded inst. requires too many bits to specify the three address.

Two-Address Inst's:

Each address field can specify either a processor register or a memory word.

	MOV	R1, A	R1 ← M[A]
	ADD	R1, B	R1 ← R1 + M[B]
May 22 Tuesday	MOV	R2, C	R2 ← M[C]
	ADD	R2, D	R2 ← R2 + M[D]
	MUL	R1, R2	R1 ← R1 * R2
	MOV	X, R1	M[X] ← R1

One Address Inst's:

One address instructions use an implied accumulator (AC) register for all data manipulation.

LOAD	A	AC ← M[A]
ADD	B	AC ← AC + M[B]
STORE	T	M[T] ← AC
LOAD	C	AC ← M[C]
ADD	D	AC ← AC + M[D]
MUL	T	AC ← AC * M[T]
STORE	X	M[X] ← AC

May 23 Wednesday All operations are done b/w AC register and memory operand. T is the address of temporary memory location required for storing the intermediate result.

Zero-Address Inst:

A stack-organized computer does not use an address field for the instruction ADD and MUL.

PUSH	A	TOS ← A
PUSH	B	TOS ← B
ADD		TOS ← (A+B)
PUSH	C	TOS ← C
May 24 Thursday PUSH	D	TOS ← D
ADD		TOS ← (C+D)
MUL		TOS ← (C+D)*(A+B)
POP	X	M[X] ← TOS

To evaluate arithmetic expressions in a stack computer, it is necessary to convert the expression into Reverse Polish Notation.

May 25 Friday

Subroutine Call and Return:

A subroutine is a self-contained sequence of instructions that performs a given task. During the execution of a program, a subroutine may be called to perform its function many times in the main program. Each time a subroutine is called, a branch is executed to the beginning of the subroutine to start executing its set of instructions. After the subroutine has been executed, a branch is made back to the main program.

The instruction that transfers program control to a subroutine is known by different

May 26 Saturday

name like call subroutine, jump to subroutine, branch to subroutine or branch and save address. A call subroutine instruction consist of an operation code together with an address that specifies the beginning of the subroutine. The last instruction of every subroutine is called return from the subroutine, transfers the return address from the temporary location into the program counter.

A subroutine call is implemented with the

May 27 Sunday

following micro operation:

SP ← SP − 1	Decrement the stack pointer.
M[SP] ← PC	Push content of PC onto the stack.

May 28 Monday

PC ← effective address T/F control to the subroutine.

The instruction that returns from the last subroutine is implemented by the micro operation;

PC ← M[SP] Pop stack and T/F to PC
SP ← SP+1 Increment stack pointer.

Program Interrupt:-

Program interrupt refers to the transfer of program control from a currently running program to another service program as a result of external or internal generated

May 29 Tuesday

request. The control returns to the original program after the service program is executed.

The interrupt procedure is quite similar to a subroutine call except for three points:

① The interrupt is usually initiated by an internal or external signal rather than from the execution of an instruction (except for s/w interrupt).

② The address of the interrupt service program is determined by h/w rather than from the address field of an instruction.

③ An interrupt procedure stores all the

MAY 2012

May 30 Wednesday Information necessary to define the state of the CPU rather than storing only the program counter.

PSW:- The collection of all status bit conditions in the CPU is sometimes called a program status word or PSW.

Types of Interrupts:

1. External interrupts
2. Internal "
3. Software "

May 31 Thursday

1. External interrupts come from I/O devices.
   e.g.:- I/O device requesting T/F of data, I/O device finished T/F of data, power failure.
2. Internal Interrupts occur from illegal or wrong instruction. Internal interrupts are also called traps.
   Examples:- register overflow, attempt to divide by zero, an invalid operation code

Note stack overflow etc.

③ Software Interrupts:

A software interrupt is initiated by executing an instruction. S/w interrupt is a special call instruction that behaves like an interrupt rather than a subroutine call.