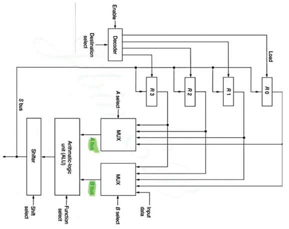
Bus Organization

When a large number of registers are included in a processor, it is the most efficient way to connect them through common buses.

- Register connected to two MUX to form input buses A and B.
- The selection lines of each MUX select one register for the particular bus.
- The A and B buses are applied to a common ALU.
- The function selected in the ALU determines the particular operation that is to be performed.
- The shift micro-operations are implemented in the shifter.
- The result of the micro-operation goes through the output bus S into the inputs of all registers.
- The destination register that receives the information from the output bus is selected by a decoder.
- When enabled, this decoder activates one of the register load inputs to provide a transfer path between the data on the S bus and the inputs of the selected destination register.
- The output bus S provides the terminals for transferring data to an external destination.
- One input of MUX A or B can receive data from the outside
- The control unit that supervises the processor bus system directs the information flow through the ALU by selecting the various components in the unit.



Processor with 2- port memory

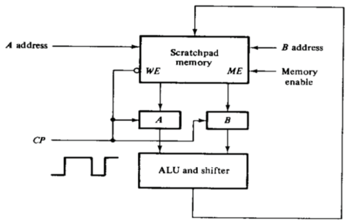
Some processor employ a 2 port memory in order to **overcome the delay caused when reading two source registers**. A 2-port has two separate address lines to select two words of memory simultaneously. The organization of a processor unit with a 2-port scratchpad memory is shown in figure above.

In this way the two source register can be read at the same time.

Memory has 2 sets of addresses. One for port A and the other for Port B.Data from any word in the memory are read into the register A by specifying an A address . Likewise data from any word in memory are read into B register by specifying a B address. The same address can be applied to the A address and B address.

When enabled by the Memory Enable (ME) input, new data can be written into the word specified byb the B address. Thus the A and B addresses specify two source registers simultaneously and the B address always specifies the destination register.

A and B registers are latches that accepts new information as the clock pulse CP is in 1 -state . When CP goes to 0, the latches are disabled and hold the information that was stored when CP was 1.



Explain zero, one, two and three address instruction with an example for each.

Three -address instruction format

This instruction format consists of three addresses along with an operation field. The three addresses include the address of the first operand, address of the second operand, address to store the result.

Format: Operation code Source1,source2, destination

Example: Add A,B,C

[C]← [A]+[B]

Two -address instruction format

This instruction format consists of two addresses along with an operation field. The two addresses include the address of the first operand, address of the second operand; the result is stored in one of the operand address.

Example: Add A, B

B← [A]+[B]

One -address instruction format

This instruction format consists of one address along with an operation field. The address is that of the first operand. The second operand and the result are stored in a CPU register called **accumulator**. A machine has only one accumulator; it need not be explicitly mentioned in the instruction.

Example: Add A

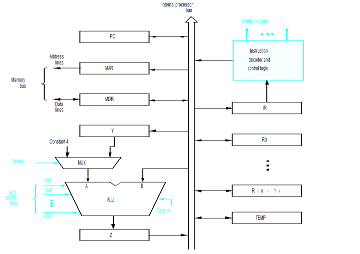
Zero -address instruction format

A **stack** is included in the CPU for performing arithmetic and logic instructions with no addresses. The operands are pushed onto the stack from memory and ALU operations are implicitly performed on the top elements of the stack.

Example: Add

Top of stack = top of stack + second top of stack

With the help of a diagram, describe the data-path inside the processor.



Processor fetches one instruction at a time and perform the operation specified.

Instructions are fetched from successive memory locations until a branch or a jump instruction is encountered.

Processor keeps track of the address of the memory location containing the next instruction to be fetched using Program Counter (PC).

Fetch the contents of the memory location pointed to by the PC. The contents of this location are loaded into the IR (fetch phase).

IR ← [[PC]]

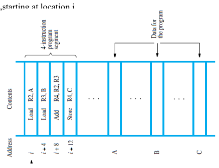
Assuming that the memory is byte addressable(4 byte long), increment the contents of the PC by 4 (fetch phase).

PC ← [PC] + 4

- PC Keeps track of execution of a program
- Contains the memory address of the next instruction to be fetched and executed.

Instruction Execution and Straight-Line Sequencing

Let's consider task C = A + B, implemented as C←[A] + [B]. Figure 8 shows a possible program segment for this task as it appears inthe memory of a computer. We assume that the word length is 32 bits and the memory isbyte-addressable. The four instructions of the program are in successive word locations,starting at location i



Since each instruction is 4 bytes long, the second, third, and fourth instructions are at addresses i + 4, i + 8, and i + 12. For simplicity, we assume that a desired memory address can be directly specified in Load and Store instructions, although this is not possible if a full 32-bit address is involved.

Let us consider how this program is executed.

- To begin executing a program, the address of its first instruction (i in our example) must be placed into the PC.
- Then, the processor control circuits use the information in the PC to fetch and execute instructions, one at a time, in the order of increasing addresses. This is called straight-line sequencing.
- During the execution of each instruction, the PC is incremented by 4 to point to the next instruction. Thus, after the Store instruction at location i + 12 is executed, the PC contains the value i + 16, which is the address of the first instruction of the next program segment.

Executing a given instruction is a two-phase procedure. In the first phase, called instruction fetch, the instruction is fetched from the memory location whose address is in the PC. This instruction is placed in the instruction register (IR) in the processor.

At the start of the second phase, called instruction execute, the instruction in IR is examined to determine which operation is to be performed. The specified operation is then performed by the processor. This involves a small number of steps such as fetching operands from the memory or from processor registers, performing an arithmetic or logic operation, and storing the result in the destination location.

At some point during this two-phase procedure, the contents of the PC are advanced to point to the next instruction. When the execute phase of an instruction is completed, the PC contains the address of the next instruction, and a new instruction fetch phase can begin.

EXECUTION OF A COMPLETE INSTRUCTION

Consider the instruction **Add (R3),R1** which adds the contents of a memory location pointed to by R3 to register R 1.

Executing this instruction requires the following actions:

1. Fetch the instruction.
2. Fetch the first operand (the contents of the memory location pointed to by R3).
3. Perform the addition.
4. Load the result into RI.

Instruction execution proceeds as follows.

Step 1: The instruction fetch operation is initiated by loading the contents of the PC into the MAR and sending a Read request to the memory. The Select signal is set to Select4, which causes the multiplexer MUX to select the constant 4. This value is added to the operand at input B, which is the contents of the PC, and the result is stored in register Z.

Step 2: The updated value is moved from register Z back into the PC, while waiting for the memory to respond.

Step 3:The word fetched from the memory is loaded into the IR.

Step	Action
1	PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
2	Z_{out}, PC_{in}, Y_{in}, WMFC
3	MDR_{out}, IR_{in}
4	$R3_{out}$, MAR_{in}, Read
5	$R1_{out}$, Y_{in}, WMFC
6	MDR_{out}, SelectY, Add, Z_{in}
7	Z_{out}, $R1_{in}$, End

Figure 16

*(Steps 1 through 3 constitute the instruction **fetch phase**, which is the same for all instructions.)*

Step 4: The instruction decoding circuit interprets the contents of the IR. This enables the control circuitry to activate the control signals for *steps 4 through 7, which constitute the **execution phase***. The contents of register R3 are transferred to the MAR in step 4, and a memory read operation is initiated.

Step 5: the contents of R1 are transferred to register Y, to prepare for the addition operation.

Step 6: When the Read operation is completed, the memory operand is available in register MDR, and the addition operation is performed. The contents of MDR are gated to the bus, and thus also to the B input of the ALU, and register Y is selected as the second input to the ALU by choosing SelectY.

Step 7: The sum is stored in register Z, and then transferred to R1. The End signal causes a new instruction fetch cycle to begin by returning to step 1.

This discussion accounts for all control signals in Figure 7.6 except Yin in step 2. There is no need to copy the updated contents of PC into register Y when executing the Add instruction. But, in Branch instructions the updated value of the PC is needed to compute the Branch target address.

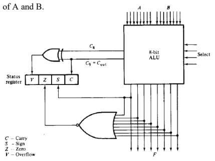
Status Registers

The relative magnitude of two numbers may be determined by subtracting one number from the other andthen checking certain bit conditions in the resultant difference. This status bit conditions (often calledcondition-code bits or flag bits) are stored in a status register.

Status register is a 4 bit register. The four bits are C (carry), Z (zero),S (sign) and V (overflow).These bits are set or cleared as a result of an operation performed in the ALU.

- Bit C is set if the output carry of an ALU is 1.
- Bit S is set to 1 if the highest order bit of the result in the output of the ALU is 1.
- Bit Z is set to 1 if the output of the ALU contains all O's.
- Bit V is set if the exclusive —OR of carries C8 and C9 is 1, and cleared otherwise. This is the condition for overflow when the numbers are in signed 2's complement representation. For an 8 bit ALU, V is set if the result is greater than 127 or less than -128.

After an ALU operation, status bits can be checked to determine the relationship that exist between thevalues of A and B.



If bit V is set after the addition two signed numbers, it indicates an overflow condition. If Z is set after anexclusive OR operation, it indicates that A=B. A single bit in A can be checked to determine if it is 0 or 1by masking all bits except the bit in question and then checking the Z status bit.

Relative magnitudes of A and B can be checked by compare operation. If A-B is performed for twounsigned binary numbers, relative magnitudes of A and B can be determined from the values transferred tothe C and Z bits. If Z=1,we knows that A=B, since A-B=0. If Z=0, then we know that A is not equal to B.

Design of Arithmetic Logic Unit

The logic circuit can be combined with the arithmetic circuit to produce one arithmetic logic unit. Selection variables S_1 and S_0 can be made common to both sections provided, we are using a third selection variable S_2, to differentiate between the two.

Design steps

1. Design the arithmetic section independent of the logic section.
2. Determine the logic operations obtained from the arithmetic circuit in step 1, assuming that the input carries to all stages are 0.
3. Modify the arithmetic circuit to obtain the required logic operations.

The third step in the design is not a straightforward procedure and requires a certain amount of ingenuity on the part of the designer. There is no guarantee that a solution can be found or that the solution uses the minimum number of gates. The example presented here demonstrates the type of logical thinking sometimes required in the design of digital systems.

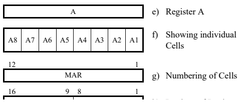
The final ALU is shown in figure below Only the first two stages are drawn, but the diagram can be easily extended to more stages. The inputs to each full-adder circuit are specified by the Boolean functions:

INTER REGISTER TRANSFER

Computer registers are designated by capital letters (sometimes followed by numerals) to denote the function of the register. [Example: R1 - Processor Register, MAR - Memory Address Register (holds an address for a memory unit), PC - Program Counter, IR - Instruction Register, SR: Status Register].The cells or flipflops of n-bit register are numbered in sequence from1 to n (from 0 to n-1) starting either from left or from right

The register can be represented in 4 ways:

- Rectangular box with name of the register inside,
- The individual cells is assigned a letter with a subscript number,
- The numbering of cells from right to left can be marked on top of the box,
- 16 bit register is partitioned into 2 parts , bits 1 to 8 are assigned the letter L(for low) and bits 9 to 16 are assigned the letter H(for high)



Registers can be specified in a register transfer language with a declaration statement. For example: Registers in the above figure can be defined with declaration statement such as

DECLARE REGISTER A(8), MBR(12), PC(16)

DECLARE SUBREGISTER PC(L) = PC(1-8), PC(H) = PC(9-16).

Information transfer from one register to another is described by a replacement operator: A ← B. This statement denotes a transfer of the content of register B into register A and this transfer happens in one clock cycle. After the operation, the content of the B (source) does not

What is Memory Transfer in Computer Architecture?

The transfer of data from a memory word to the external environment is known as a read operation. The read operation in memory transfer is represented as the transfer of data from the address register (AR) with the selected word M for the memory into the memory buffer register (MBR).

[AR]M MBR=Read Operation

The control signal of the read operation starts the read operation. The read operation statement generates the data transfer from the chosen memory register M into the MBR.

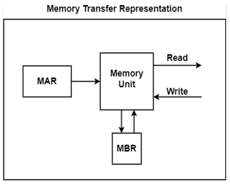
The transfer of new data to be saved into the memory is known as the **write operation**. The memory transfer in the write operation is described as the transfer of data from the memory buffer register (MBR) to the address register (AR) with the chosen word M for the memory.

MBR M [AR] =Write Operation.

The control signal of the write operation starts the write operation. The write operation statement generates the data transfer from the MBR into the chosen memory register through the address shows in the memory M [AR].

It can achieve through a read or write operation, first, the memory register (M) should be selected by a particular address.

The figure shows the memory transfer representation. It demonstrates that the memory unit can transfer the data from the memory address register and memory buffer register to implement read and write operations in the memory transfer.



The Booth Algorithm

Algorithm & Flowchart for Booth Multiplication

1. Multiplicand is placed in BR and Multiplier in QR
2. Accumulator AC, Q_{n+1} are initialized to 0
3. Sequence counter SC is initialized to n (number of bits).
4. Compare Q_n and Q_{n+1} and perform the following

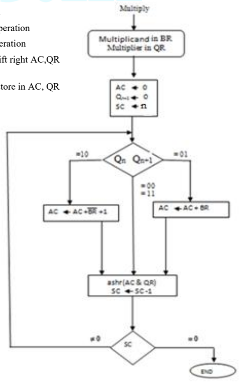
   01 –> AC=AC+BR

   10 –> AC=AC+BR'+1

   00 –> No arithmetic operation

   11-> No arithmetic operation
5. ASHR- Arithmetic Shift right AC,QR
6. Decrement SC by 1

The final product will be store in AC, QR



Multiply
Multiplicand in BR
Multiplier in QR
AC ← 0
Q_{n+1} ← 0
SC ← n
Q_n Q_{n+1}
=10
=01
=00
=11
AC ← AC+$\overline{BR}$+1
AC ← AC+BR
ashr(AC & QR)
SC ← SC-1
SC
≠0
=0
END