COA Important Questions

module 1

Part A

1. What is an instruction code?
2. What is instruction cycle
3. Write the use of PC register.
4. Write about fetch – decode – execute cycle.
5. How are memory locations addressed?(2)
6. Write about branch instructions// refer mp
7. Which are different buses
8. Define Memory-reference instruction.
9. Explain Reverse Polish Notation.

Part B

1. Explain stored program organization
2. Explain bus organization(2)
3. Explain about data transfer instructions with suitable examples. // refer mp
4. Explain how instruction is executed(2)
5. Explain general purpose registers
6. Explain types of interrupts.

Part C

1. Basic functional units of computer with diagram(2)
2. Explain Instruction cycle.

module 2

Part A

1. What is stack pointer?
2. What is the purpose of program control instructions?
3. What is a stack
4. What is Two Address Instruction

Part B

1. Explain register organization in CPU
2. Explain Register stack with figure.
3. Explain general register organization with figure.
4. Differentiate direct and indirect address.

Part C

1. Describe general register organization with the help of neat diagram.

2. (a)Explain CPU internal architecture
3. Explain Various addressing modes with example.(2)

module 3

Part A

1. What is SRAM and DRAM?
2. Write a note on primary memory.
3. What is page fault?
4. What is EPROM
5. Write a note on primary memory.

Part B

1. What is the use of auxiliary memory
2. Explain how magnetic tapes work.
3. Discuss the replacement algorithms for cache memory.
4. Explain address mapping using pages.
5. Differentiate SRAM and DRAM?
6. Explain hardware organization of associative memory
7. Different types of ROM
8. Features of PROM
9. Differentiate RAM and ROm
10. Explain cache memory

Part C

1. Explain memory hierarchy.
2. Explain virtual memory and different mapping techniques

module 4

part A

1. What is MISD?
2. What is parallel computing

part B

1. Explain multi processing systems
2. Features of SISD
3. Flynn's Classification of Computers

part C

1. Write a short note on parallel system

module 5

part A

2. Advantages of pipelining
3. What is pipelining

part B

1. What is space - time diagram for a pipeline?
2. Differentiate RISC and CISC
3. How instructions can be pipelined

part C

1. Explain Instruction pipeline with example and neat diagram.
2. Explain design of Arithmetic pipeline
3. Explain different types of array processors. What are the advantages provided by array
4. processors?(2)

Reverse Polish notation (RPN)

Reverse Polish notation (RPN) is a method for conveying mathematical expressions without the use of separators such as brackets and parentheses. In this notation, the operators follow their operands, hence removing the need for brackets to define evaluation priority. The operation is read from left to right but execution is done every time an operator is reached, and always using the last two numbers as the operands. This notation is suited for computers and calculators since there are fewer characters to track and fewer operations to execute.

Reverse Polish notation is also known as **postfix** notation.

Arithmetic Expression Evaluation

The stack organization is very effective in evaluating arithmetic expressions. Expressions are usually represented in what is known as **Infix notation**, in which each operator is written between two operands (i.e., A + B). With this notation, we must distinguish between (A + B)*C and A + (B * C) by using either parentheses or some operator-precedence convention. Thus, the order of operators and operands in an arithmetic expression does not uniquely determine the order in which the operations are to be performed.

1. **Polish notation (prefix notation) –**
   It refers to the notation in which the operator is placed before its two operands . Here no parentheses are required, i.e.,
   ```
   +AB
   ```
2. **Reverse Polish notation(postfix notation) –**
   It refers to the analogous notation in which the operator is placed after its two operands. Again, no parentheses is required in Reverse Polish notation, i.e.,
   ```
   AB+
   ```

Stack organized computers are better suited for post-fix notation then the traditional infix ntation. Thus the infix notation must be converted to the post-fix notation. The conversion from infix notation to post-fix notation must take into consideration the operational hierarchy.

There are 3 levels of precedence for 5 binary operators as given below:

```
Highest: Exponentiation (^)

Next highest: Multiplication (*) and division (/)

Lowest: Addition (+) and Subtraction (-)
```

For example –

```
Infix notation: (A-B)*[C/(D+E)+F]

Post-fix notation: AB- CDE +/F +*
```

Here, we first perform the arithmetic inside the parentheses (A-B) and (D+E). The division of C/(D+E) must done prior to the addition with F. After that multiply the two terms inside the parentheses and bracket.

Now we need to calculate the value of these arithmetic operations by using stack.

The procedure for getting the result is:

1. Convert the expression in Reverse Polish notation(post-fix notation).
2. Push the operands into the stack in the order they appear.

3. When any operator encounter then pop two topmost operands for executing the operation.
4. After execution push the result obtained into the stack.
5. After the complete execution of expression the final result remains on the top of the stack.

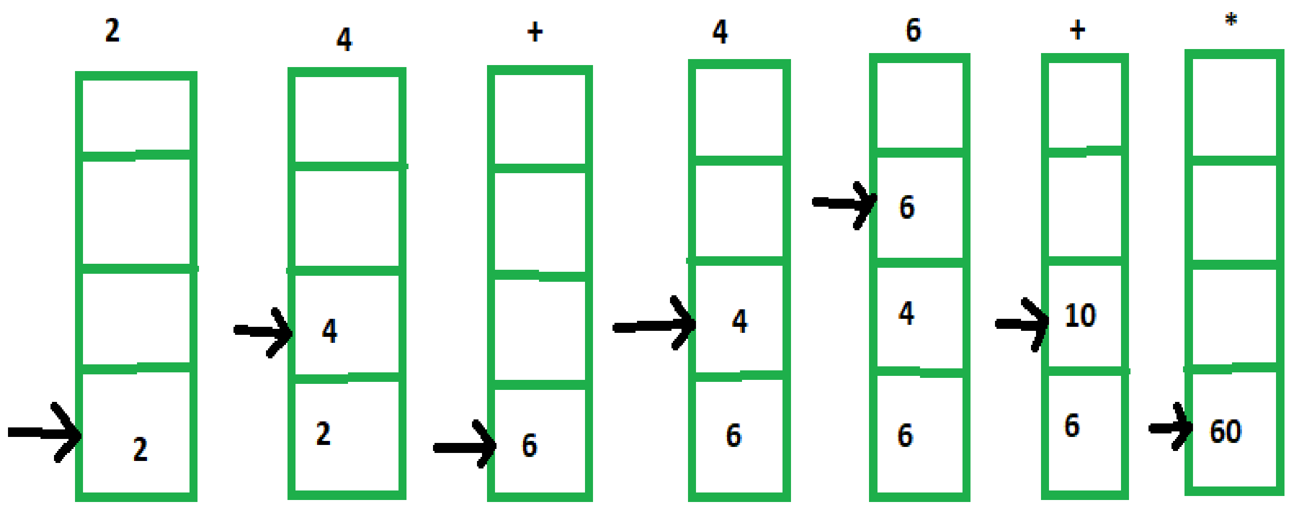
For example –

Infix notation: (2+4) * (4+6)

Post-fix notation: 2 4 + 4 6 + *

Result: 60

The stack operations for this expression evaluation is shown below:



Stack operations to evaluate (2+4)*(4+6)