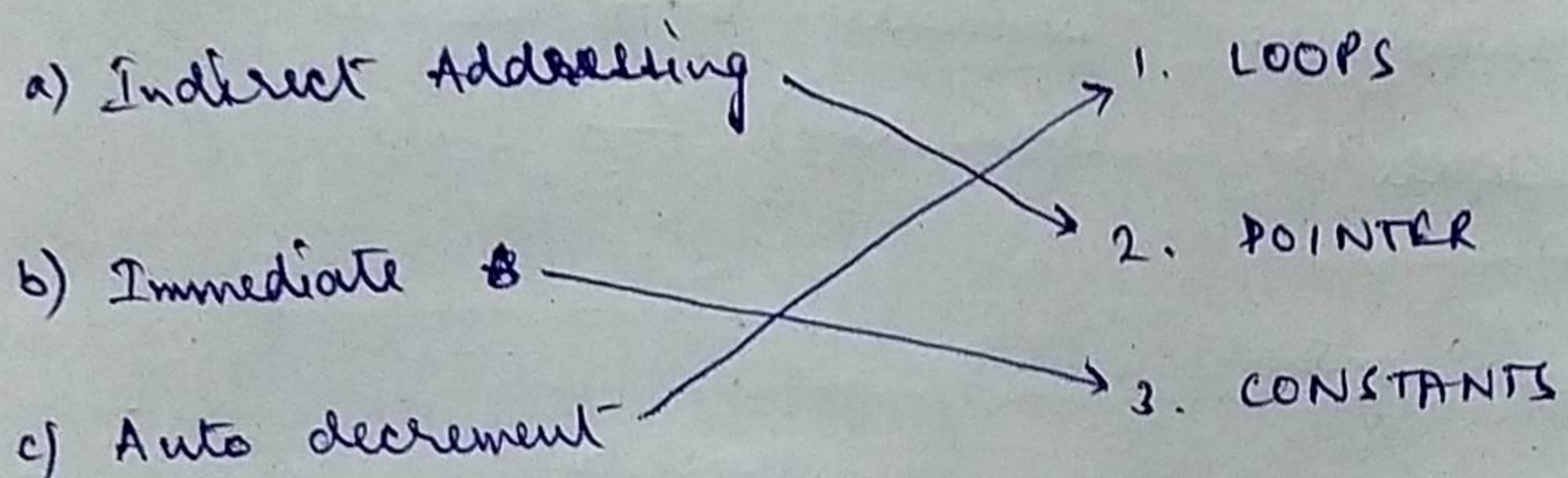
GATE 2000

1. Match the following:



GATE 2002

Q. In absolute addressing mode : (absolute ≡ direct)

a) The operand is inside the instruction

b) The address of operand is inside instruction

c) Register containing address of operand is specified in instructions.

d) Location of operand is implicit.

GATE 2009

8. which modes are suitable for program relocation at run time?

a) Absolute (b) Base (c) Relative d) Indirect

GATE 2001

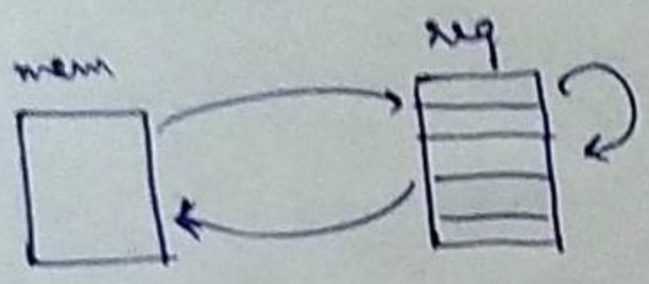
8. Indirect → Array implementation

Indexed → Relocatable code

Base register → Passing array as parameter.

TYPES OF INSTRUCTIONS

- DATA TRANSFER INSTRUCTIONS



- Data Manipulation Instruction (to manipulate & transform data)
  - arithmetic
  - logical
  - shift
- Program control
  - Decision based instruction.
    (if, for, while)

DATA TRANSFER INSTRUCTIONS

Responsible for moving data from various memory, registers, I/O devices.

1) Load → Load from memory to register
2) Store → (Register → Memory)
3) Exchange → (Register ⇄ Memory) (swap)
4) Input & output → B/w register & I/O
5) PUSH & POP → used in stack (memory or register)

ARITHMATIC Instructions :-

1) Increment & Decrement
2) Add & subtract } provided by almost all systems
3) Multiple & division.
4) Add with carry
5) Subtract with borrow

> NOTE : Each of these instructions have several micro operations involved before them.

6) Negate → Complement of data.

Logical Instructions (Also called bit triggering operation)

1) AND } Provided in all systems
2) OR
3) CLEAR → changing all bits of register to 0.
4) COMPLEMENT → done using Ex OR (another operation)
5) Exclusive OR → Provided by many
6) Clear / set carry → set carry to 0 or 1.
7) Complement carry
8) Enable / disable Interrupt

SHIFT INSTRUCTIONS

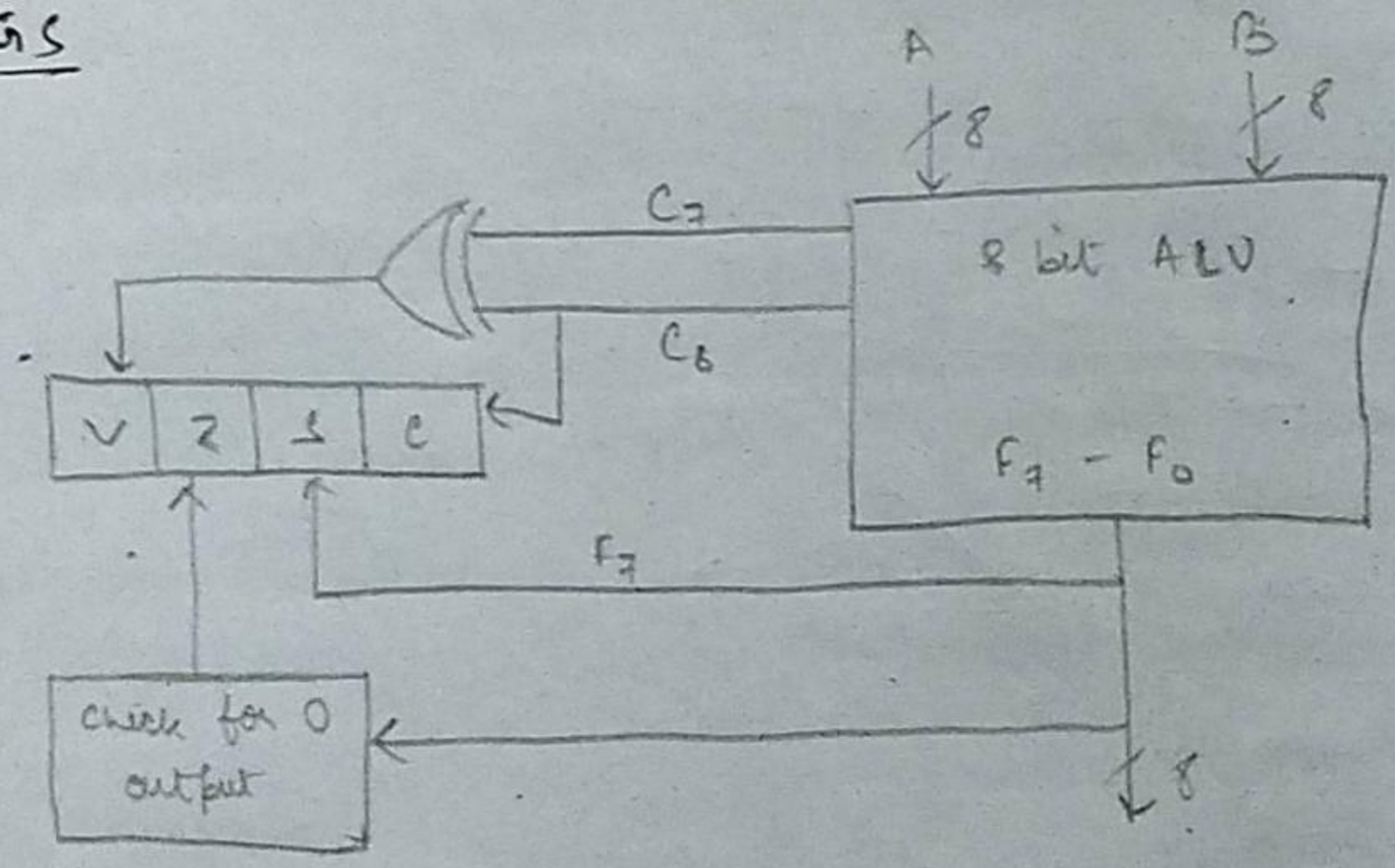
1) Logical shift Right / left → shift all bits to right / left & M.S.B will have 0.
2) Arithmatic shift right / left → M.S.B or sign bit remains same. All other bits get shifted. The empty place gets filled by sign bit.
   Same as logical left
3) Rotate Left / Right → shift considering register as a circle.
4) Rotate Through carry → & consider register + carry and shift as a circle.

PROGRAM CONTROL INSTRUCTION

(To change flow of execution from sequential to non sequential)

1. Branch → Conditional / Non-conditional. (Direct mode)
2. JUMP → Same as branch. (Indirect mode)
3. SKIP → skip the next step.
4. CALL → Call function / procedure
5. RETURN → Return the control
6. COMPARE → Compare A & B and apply instruction
7. TEST → similar to compare.

FLAGS



V → overflow flag ($C_{in} \oplus C_{out}$)

- 1 → overflow
- 0 → no overflow

Z → indicate whether result is zero or not

S → sign bit (used to identify sign of result)

C → Carry (To see if last instⁿ left a carry or not)