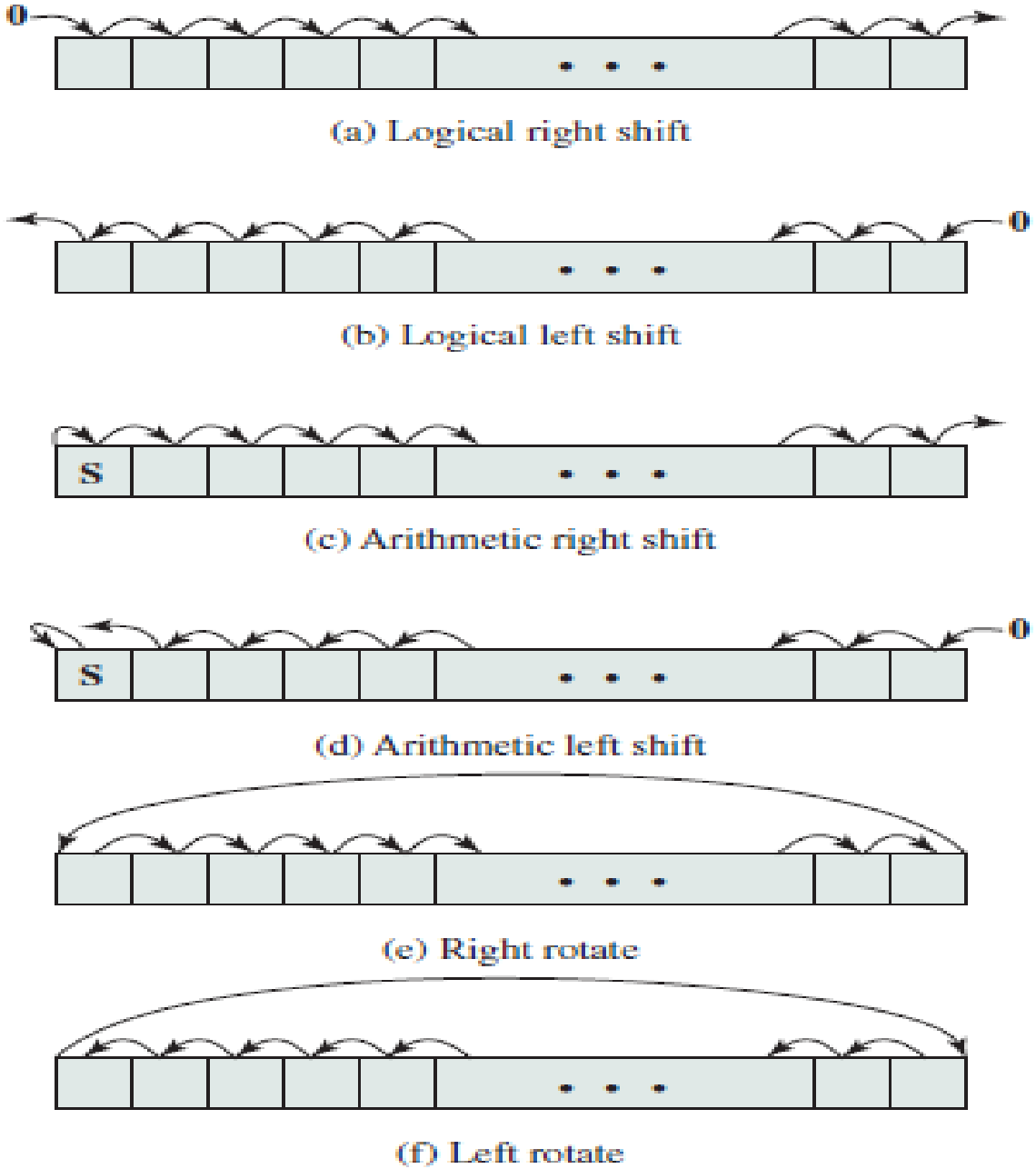
Computer Organization and Software Systems

Contact Session 7

Dr. Lucy J. Gudino

Shift and Rotate Operati



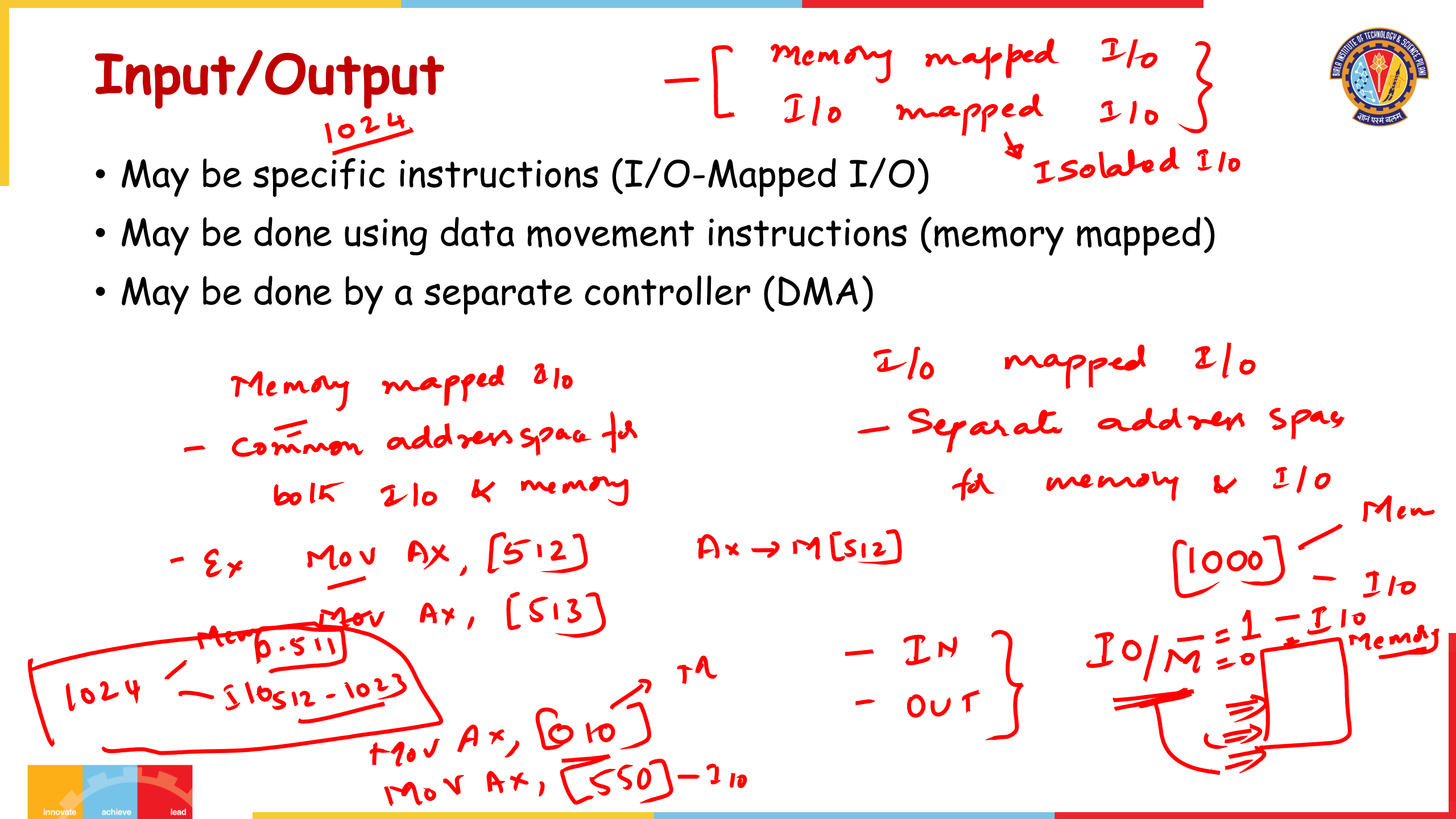
Input	Operation	Output
10101101	Logical right shift (3 bits)	10101101=> 00010101
10101101	Logical left shift (3 bits)	10101101=> **01101000**
10101101	Arithmetic right shift (3 bits)	10101101=> 11110101
10101101	Arithmetic left shift (3 bits)	10101101=> **11101000**
10101101	Right rotate (3 bits)	10101101=> **10110101**
10101101	Left rotate (3 bits)	10101101=> **01101101**

Conversion

- E.g. Binary to Decimal

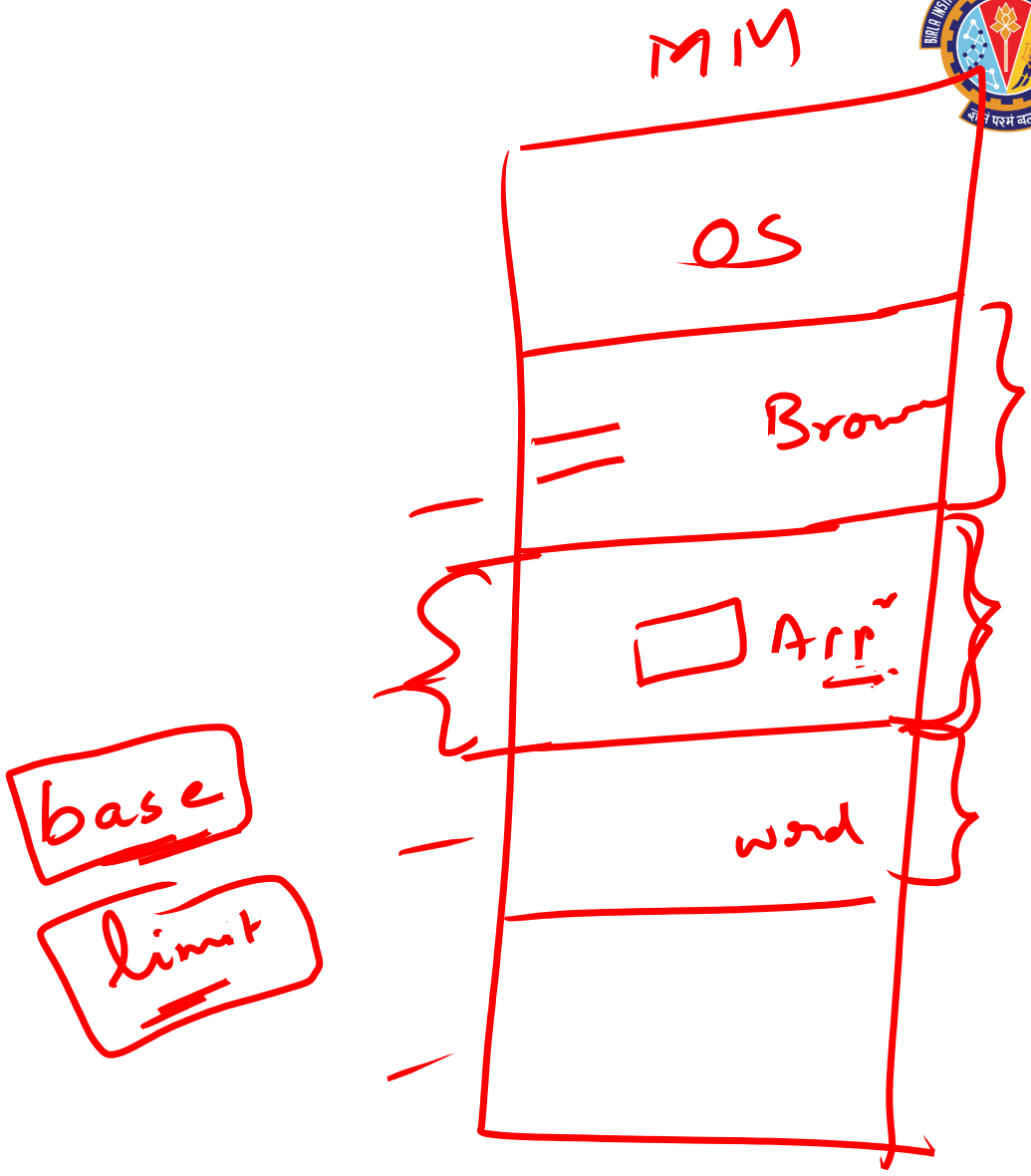
Input/Output

- May be specific instructions (I/O-Mapped I/O)
- May be done using data movement instructions (memory mapped)
- May be done by a separate controller (DMA)



Systems Control

- Privileged instructions
- CPU needs to be in specific state
  - User Mode
  - Kernel mode
- For operating systems use



Transfer of Control

- Jump / Branch (Unconditional / Conditional)
  - e.g. jump to x if result is zero
- Skip (Unconditional / Conditional)
  - skip (unconditional) : Increment to skip next instruction
  - e.g. increment and skip if zero

Conditional

ISZ Register1

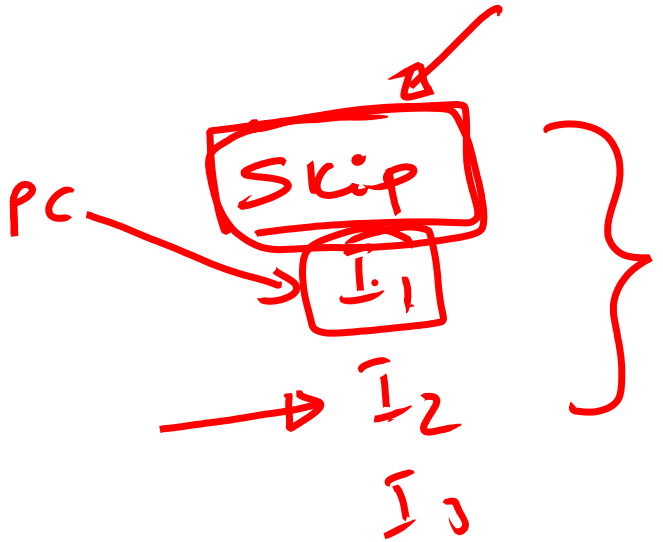
→ Branch xxxx

→ ADD A

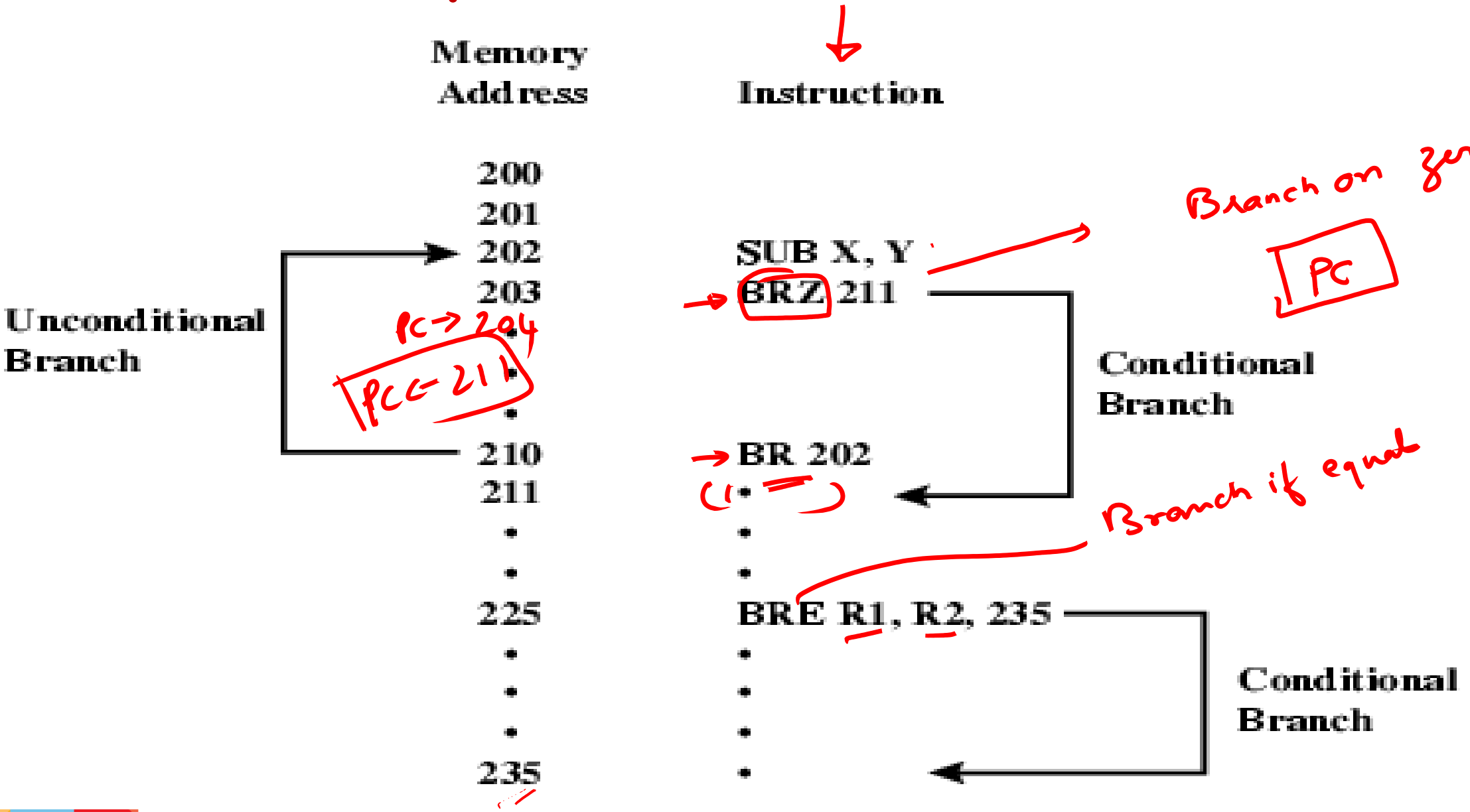
PC ←

JNZ 1000

PC ← 1000



Branch / Jump Instruction



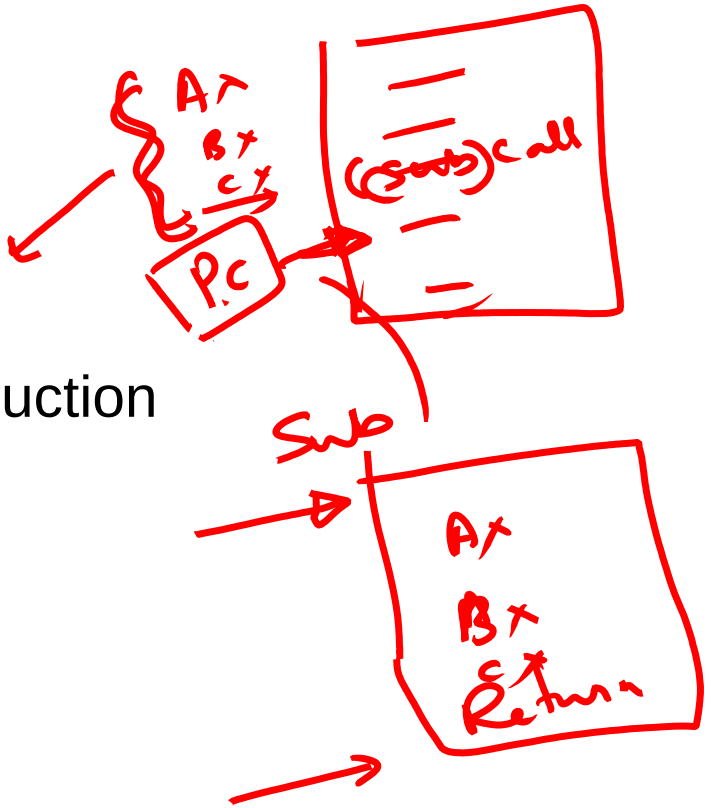
Transfer of Control

- Jump / Branch (Unconditional / Conditional)
  - e.g. jump to x if result is zero
- Skip (Unconditional / Conditional)
  - skip (unconditional) : Increment to skip next instruction
  - e.g. increment and skip if zero

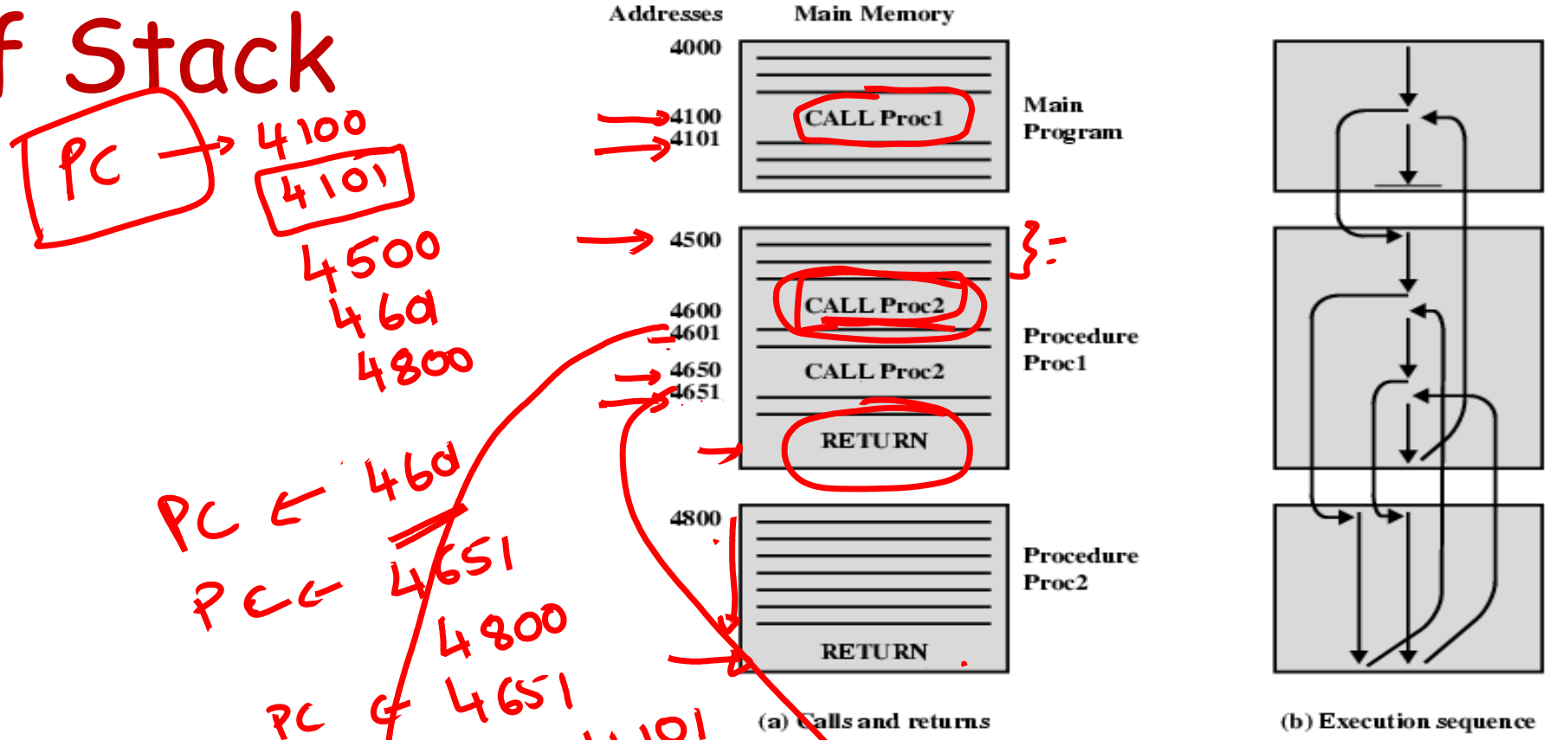
    ISZ Register1

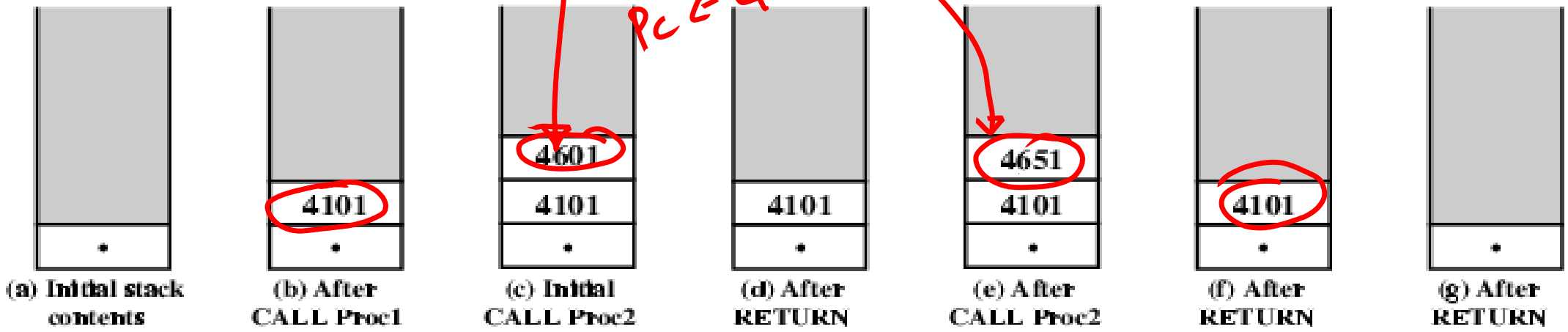
    Branch xxxx

    ADD A
- Subroutine call ✓
- interrupt call ✓



Use of Stack


Addresses
Main Memory
4000
4100
4101
CALL Proc1
Main Program
4500
CALL Proc2
4600
4601
Procedure Proc1
4650
4651
CALL Proc2
RETURN
4800
Procedure Proc2
RETURN
(a) Calls and returns
(b) Execution sequence


4101
4601
4101
4101
4651
4101
4101
(a) Initial stack contents
(b) After CALL Proc1
(c) Initial CALL Proc2
(d) After RETURN
(e) After CALL Proc2
(f) After RETURN
(g) After RETURN

Addressing Modes

- Addressing modes refers to the way in which the operand of an instruction is specified
- Types:
  - Immediate
  - Direct
  - Indirect
  - Register
  - Register Indirect
  - Displacement (Indexed)
  - Stack

Immediate Addressing

- Operand is specified in the instruction itself
- e.g. ADD #5
  - Add 5 to contents of accumulator
  - 5 is operand
- No memory reference to fetch data
- Fast
- Limited range

ADD 5

+7 to -8

Add #-5

AC ← AC + 05

ADD AX, 05

0000 → Pos :- 8 to 16

1111 → negat 8

16

$2^6 = 64$

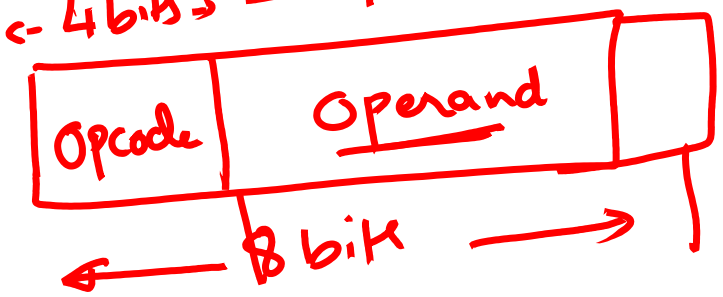
32

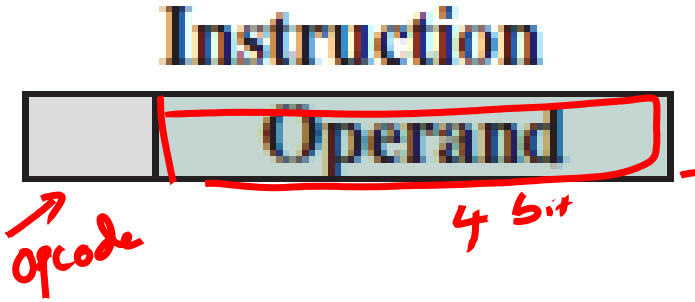
32

Signed m

One's

for(i=0; i<4; i++)





2's compln

0

7

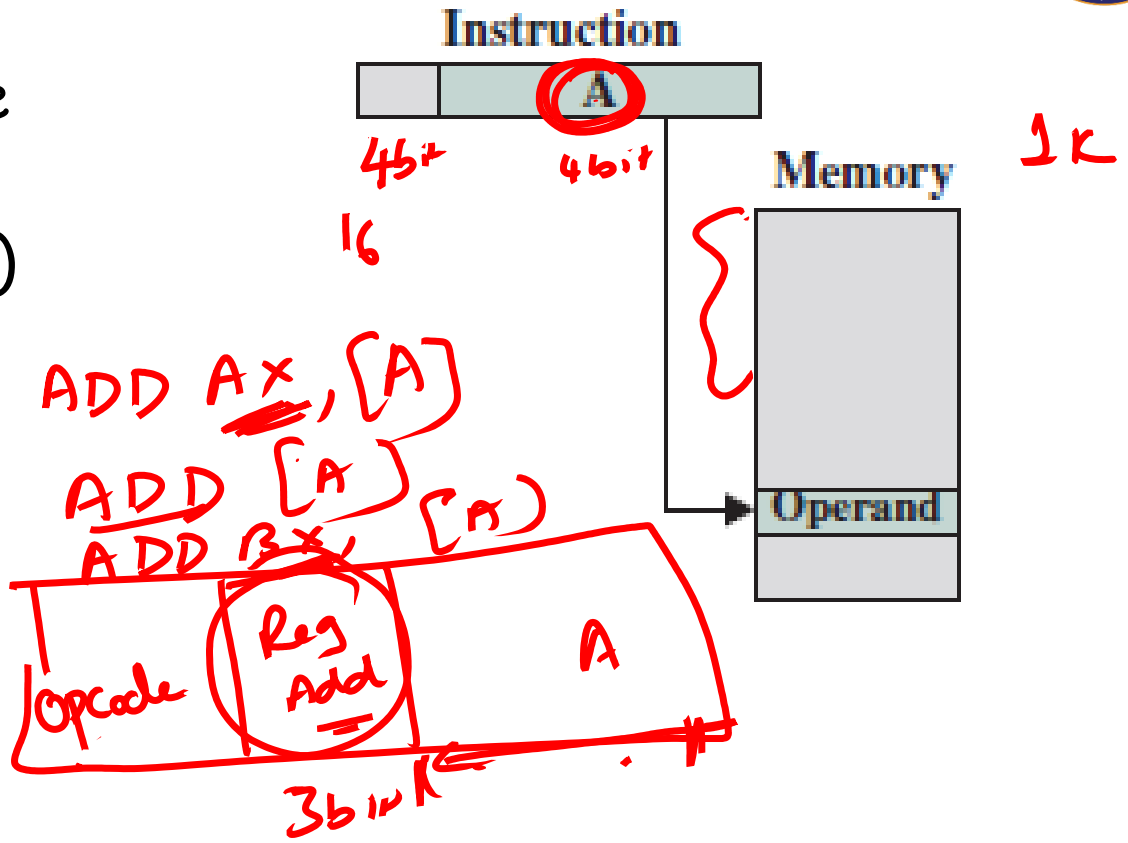
-1

-2

-8

Direct Addressing

- Address of the operand is specified in the instruction
- Effective address (EA) = address field (A)
- e.g. ADD(A)
  - Add contents of memory cell whose address is A to accumulator
  - Look in memory at address A for operand
- Single memory reference to access data
- No additional calculations to work out effective address
- Limited address space



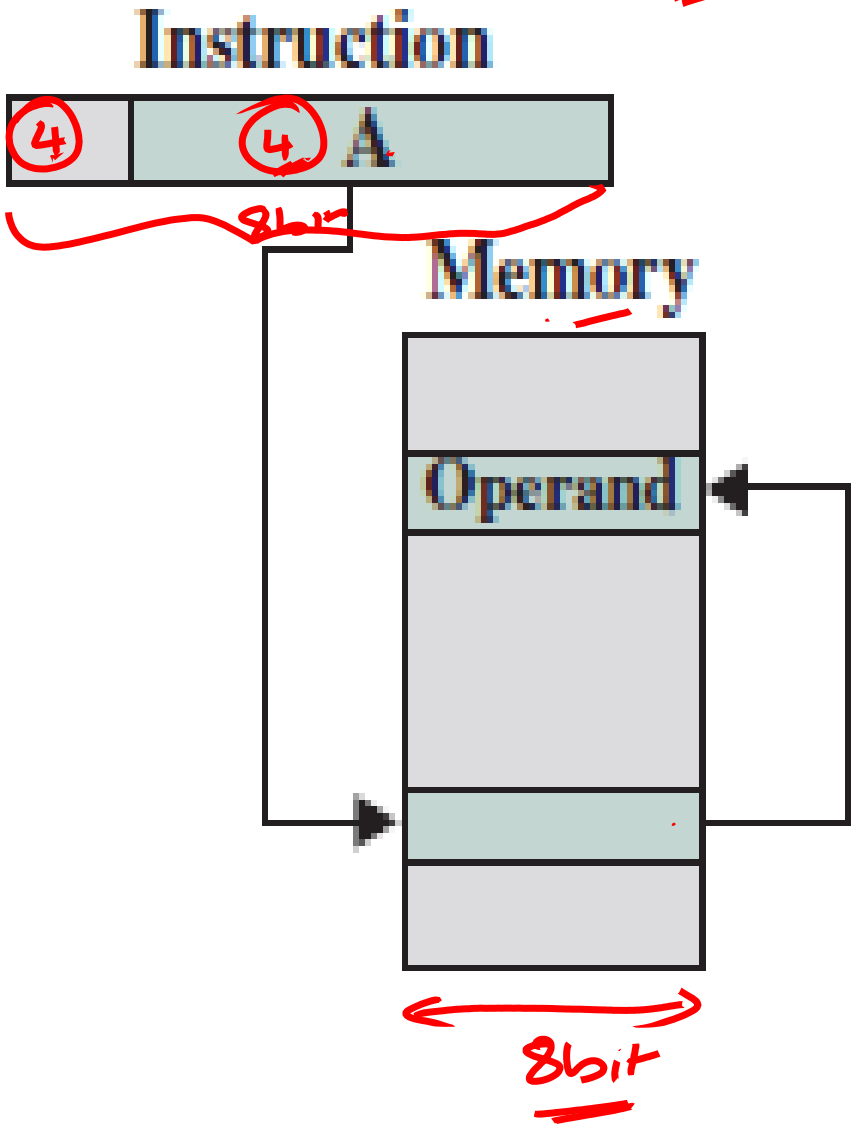
Indirect Addressing

- Memory cell pointed to by address field of the instruction contains the address of (pointer to) the operand
- EA = (A)

  - Look in A, find address and look there for operand

- e.g. ADD (A)

  - Add contents of cell pointed to by contents of A to accumulator



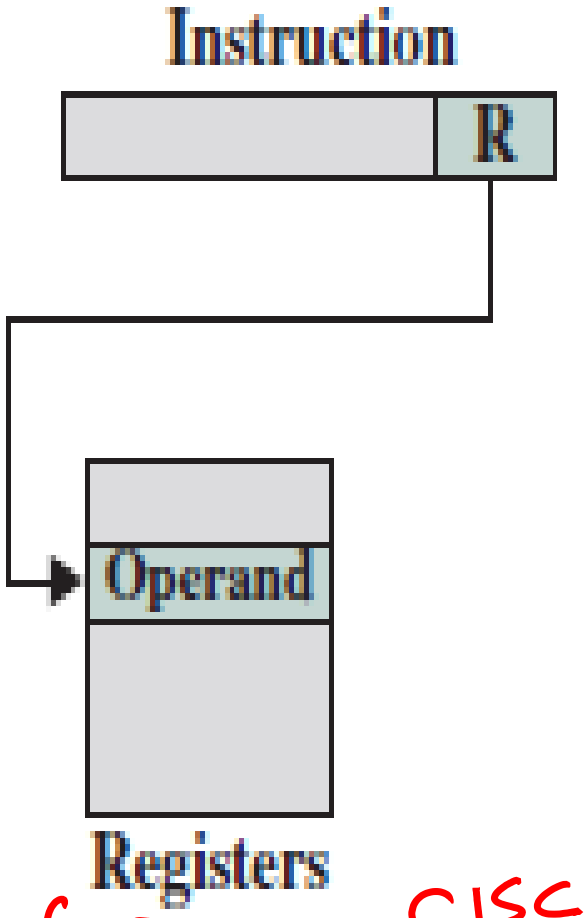
Indirect Addressing…

- Large address space
- 2^n where n = word length
- May be nested, multilevel, cascaded
  - e.g. EA = (((A)))
- Multiple memory accesses to find operand
- Slower

Register Addressing

- Operand is held in register named in address filed
- EA = R
- Limited number of registers
- Very small address field needed
  - Shorter instructions
  - Faster instruction fetch
- No memory access hence Very fast execution but very limited address space
- Multiple registers helps in improving performance
  - Requires good assembly programming or compiler writing
  - C programming : register int a;

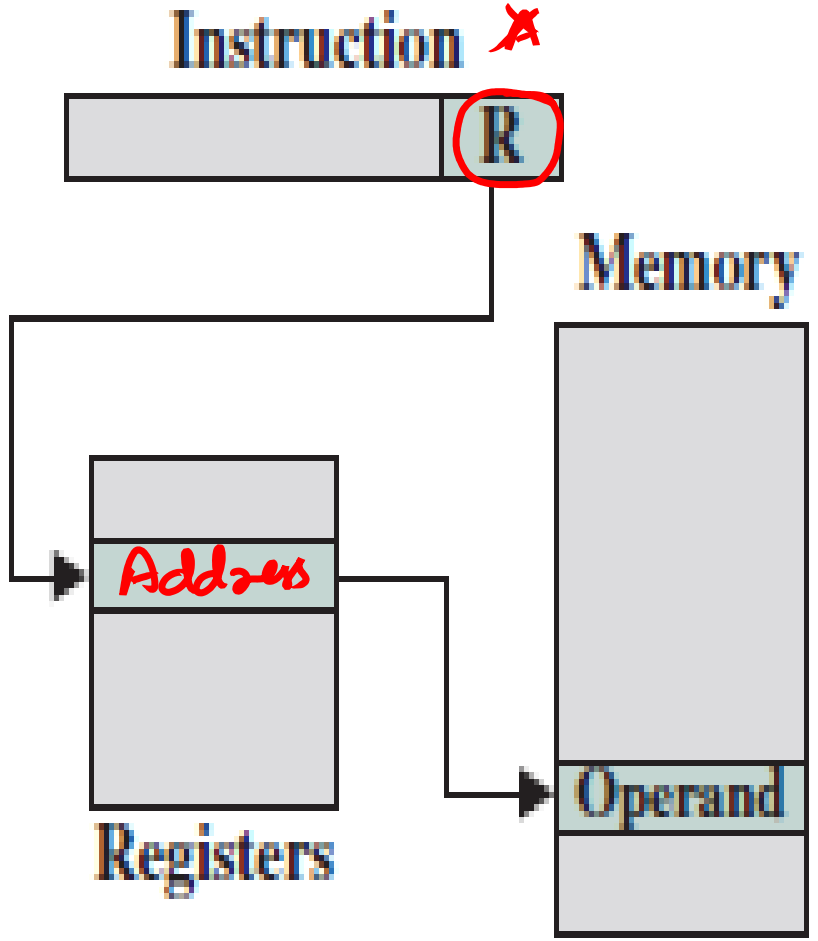
automatic
Register
extern
static



Add Ax, [A] → CISC
RISC

Register Indirect Addressing

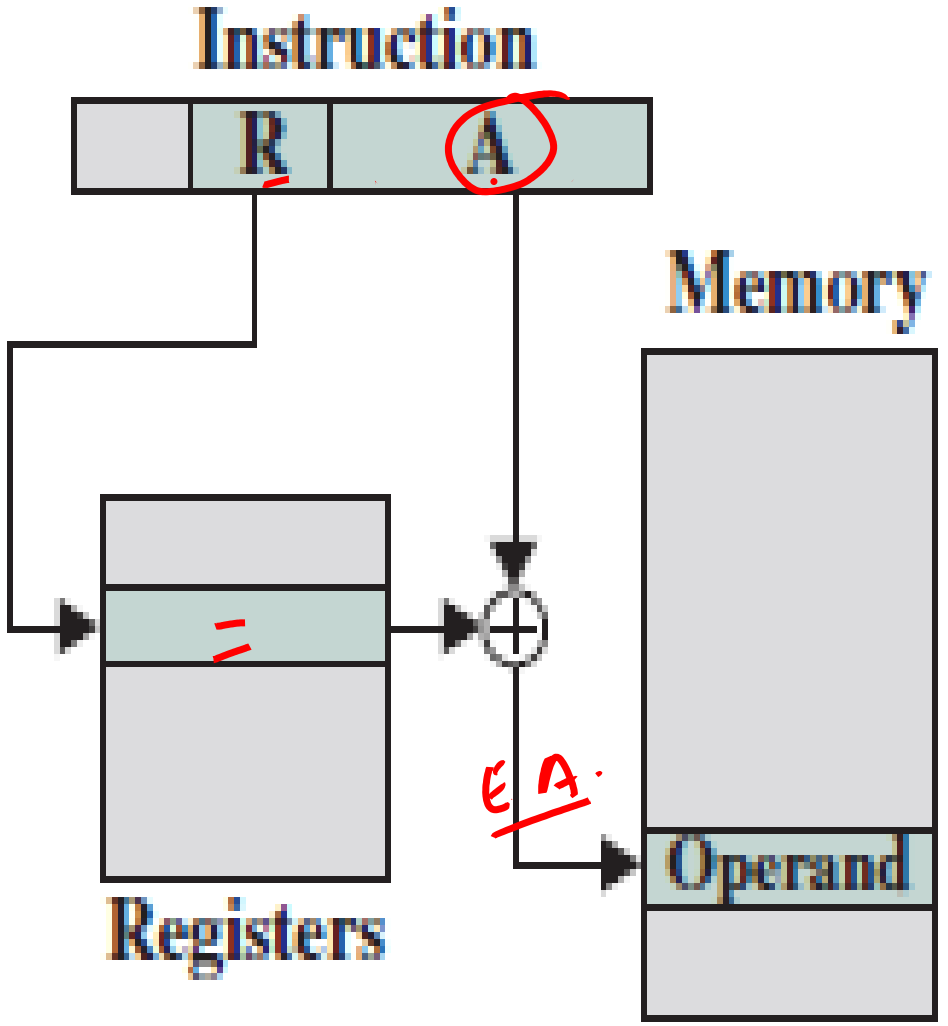
- Similar to indirect addressing
- EA = (R)
- Operand is in memory cell pointed to by contents of register R
- Large address space (2^n)
- One memory access compared indirect addressing



Direct addressing → 1 memory access
Indirect addressing → 2 "
Reg I A →

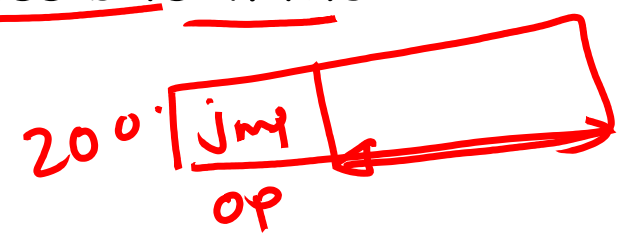
Displacement Addressing

- EA = A + (R)
- Address field hold two values
  - A = base value
  - R = register that holds displacement
  - or vice versa
- Three variants:
  - Relative addressing
  - Base register addressing
  - Indexing



Relative Addressing

- Also known as PC relative addressing
- A version of displacement addressing
- R = Program counter, PC
- EA = A + (PC)
- Relative addressing exploits the concept of locality
  - If most memory references are relatively near to the instruction being executed, then the use of relative addressing saves address bits in the instruction.



Base-Register Addressing

- The referenced register "R" contains a main memory address
- address field contains a displacement A
- R may be explicit or implicit
- e.g. segment registers in 80x86

8086

DS - Data → Data

CS - Code → Prog inst

ES - Extra → Data

SS - Stack →

Indexed Addressing

- The address field references a main memory address A
- The referenced register R contains a positive displacement from that address.
- EA = A + R
- Good for accessing arrays
  - EA = A + R
  - R++

SI - Source Index

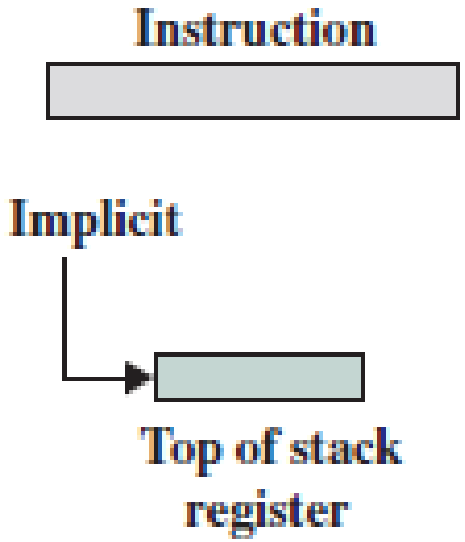
DI - Destination Index

INR SI

DCR SI

Stack Addressing

- Operand is (implicitly) on top of stack
- e.g.
  - ADD Pop top two items from stack and add, push the result on stack top



Instruction Formats

- Layout of bits in an instruction
  - Includes opcode
  - Includes (implicit or explicit) operand(s)
  - Usually more than one instruction format in an instruction set

R2000 - 32bit

8086 - 1 to 6 bytes

— fixed instruction format

Variable instruction format

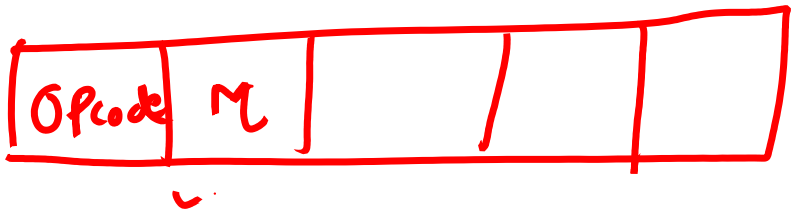
PC → PC + 4

Instruction Length

- Affected by and affects:
  - - Memory size
  - - Memory organization
  - - Bus structure
  - - CPU complexity
  - - CPU speed
- Trade off between powerful instruction repertoire and saving space

Allocation of Bits

- Number of addressing modes – mod bit
- Number of operands – 0, 1, 2, and 3
- Register versus memory
- Number of register sets –
- Address range –
- Address granularity



CISC Vs RISC

#	RISC	CISC
1.	Reduced Instruction Set Computer.	Complex Instruction Set Computer.
2.	Fixed length instructions	Variable length instructions
3.	Instructions are executed in one clock cycle.	Takes one or more clock cycle
4.	Relatively simple to design.	Complex to design.
5.	Fewer, Simple addressing modes	Many addressing modes
6.	Hardwired Control Unit	Microprogrammed Control Unit
7.	Harvard Architecture	Von-Neumann Architecture
8.	Examples: SPARC, POWER PC.	Examples: Intel architecture(x86 and x64) AMD.