University of Engineering and Technology (UET),
Peshawar, Pakistan

Lecture 9

CSE-304: Computer Organization and Architecture

BY:

Dr. Muhammad Athar Javed Sethi

Slides are taken from student resources of William Stallings website

Instruction set

- The complete collection of instructions that are a CPU can perform
- A program is a “machine code” of consecutive machine instructions
- Represented in binary form
- Usually described by assembly codes

Elements of an Instruction

- Operation code (Op code)
  - Do this
- Source Operand reference
  - To this
- Result Operand reference
  - Put the answer here
- Next Instruction Reference
  - When you have done that, do this...
  - Implicit in case of sequential execution (PC stores it)

Operand source and destination

- Memory
- CPU register
- I/O device

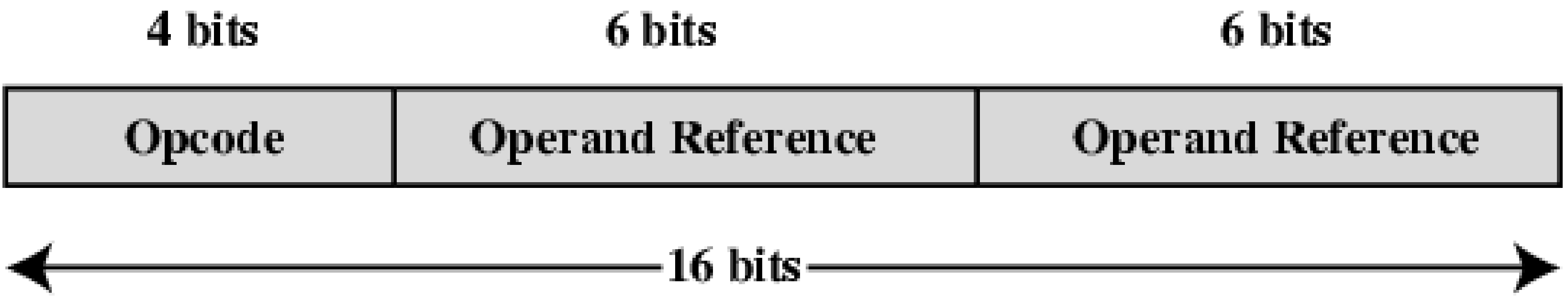
Instruction Representation

- In machine code each instruction has a unique bit pattern
- For human user (programmer) a symbolic representation is used
  - e.g. ADD, SUB, LOAD
- Operands can also be represented in this way
  - ADD R, A
  - Note: operation performed on the contents of A (not on the address itself)

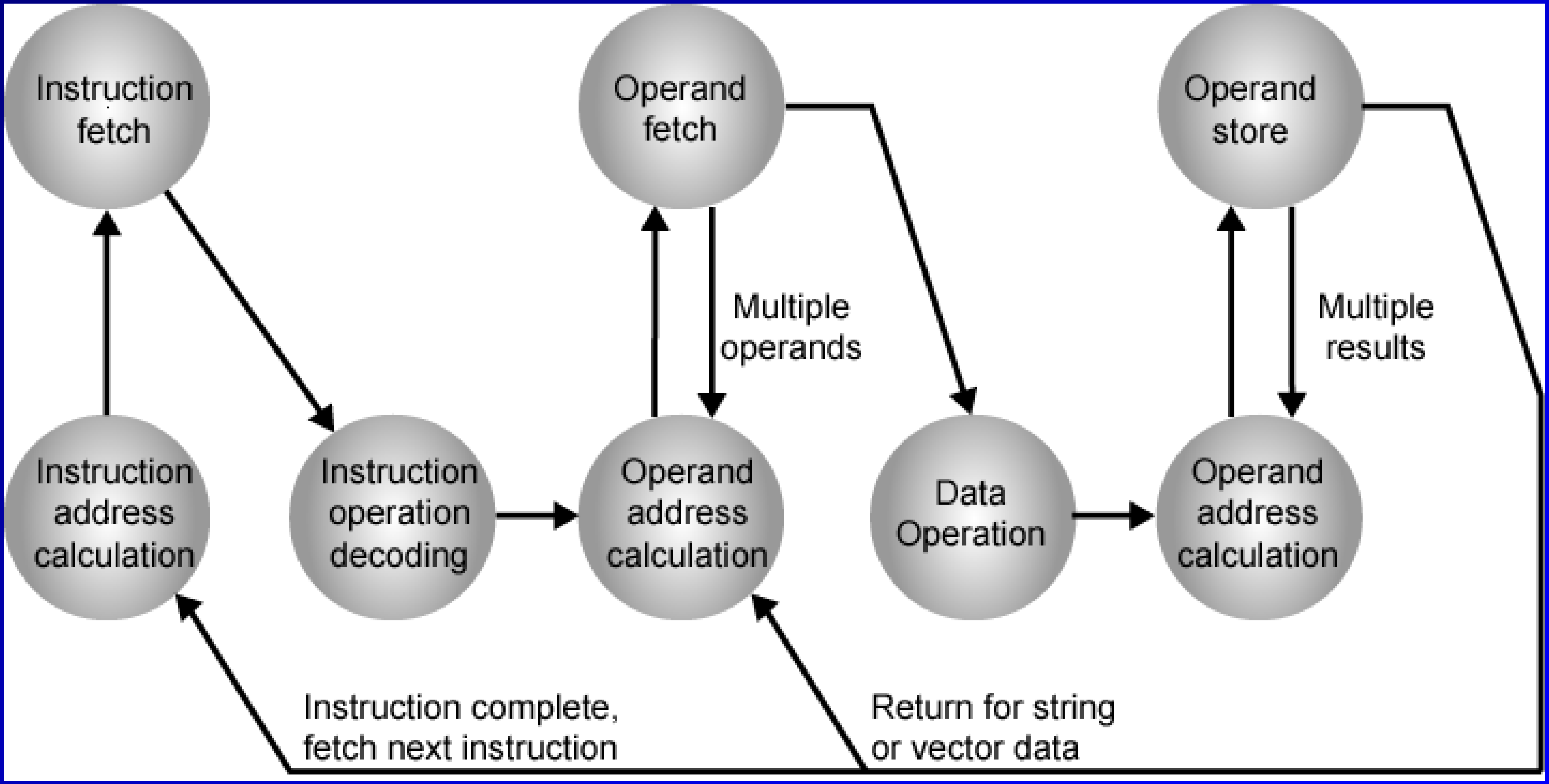
Simple Instruction Format

Characteristics:

- 2 operand instruction
  - Could be 0, 1, 2, 3 or more …
- 4 bit opcode
  - Instruction set can have at most 16 instructions
- 6 bit operands
  - limit on memory + number representation

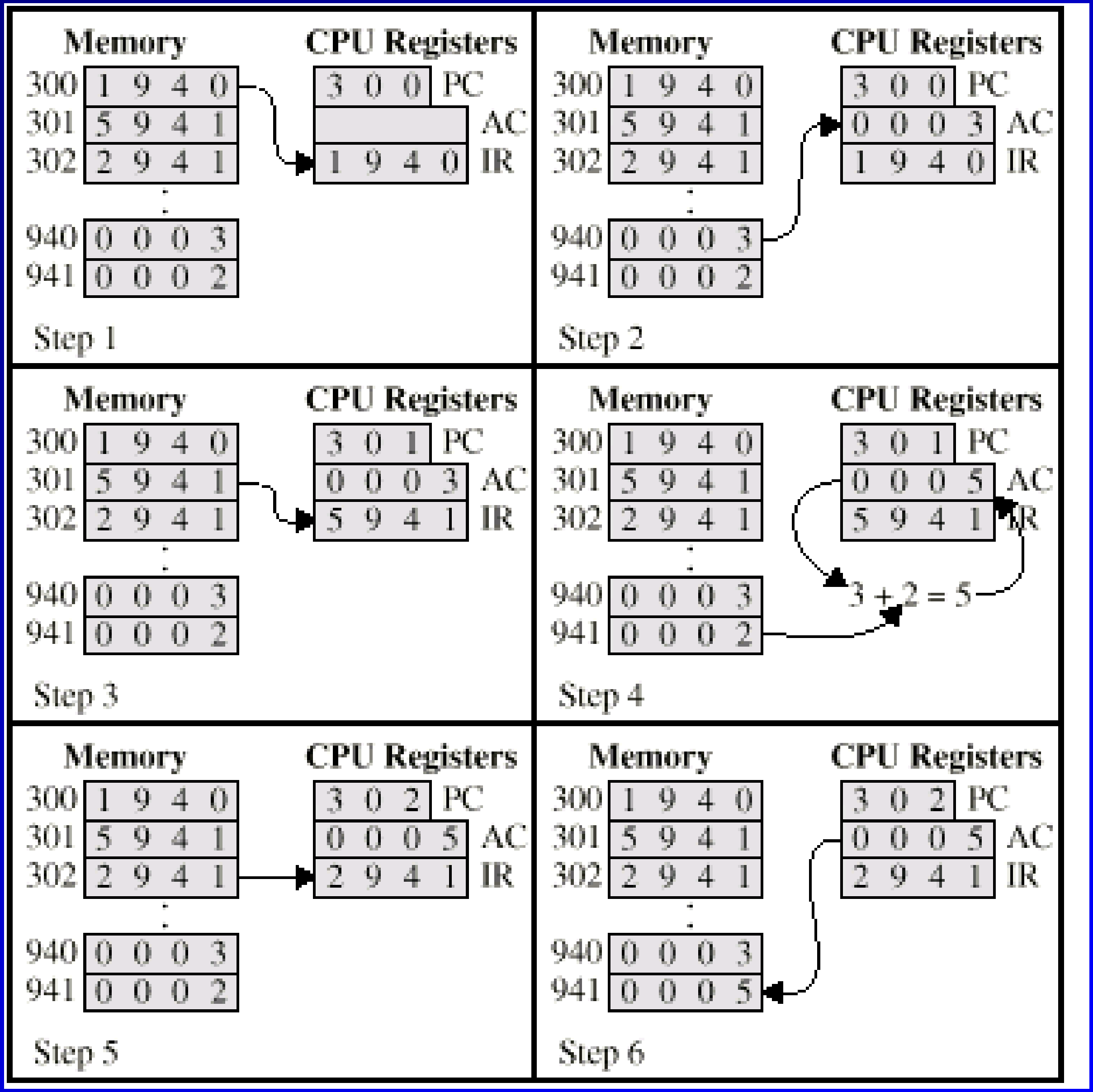


Instruction Cycle State Diagram


Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data Operation
Operand address calculation
Instruction complete, fetch next instruction
Return for string or vector data

Example of Program Execution

Fetch



Execute

Instruction Types

- Data processing
  - Arithmetic and logic instruction
- Data storage
  - Memory instructions:
  - Load from memory, Store to memory
- Data movement
- I/O operations
  - Read from I/O device, write to I/O device
- Program flow control
  - Test instructions (used e.g. in conditionals)
  - Branch instructions (used in conditionals, loops)

Number of Addresses

- 3 addresses
  - Operand 1, Operand 2, Result
  - ADD A, B, C
  - $A \longleftarrow B + C$
  - May be a forth - next instruction (usually implicit)
  - Not common
  - Needs very long words to hold everything

Number of Addresses

- 2 addresses
  - One address doubles as operand and result
  - ADD A, B
  - $A \longleftarrow A + B$
  - Reduces length of instruction
  - Requires some extra work
    - Temporary storage to hold some results

Number of Addresses

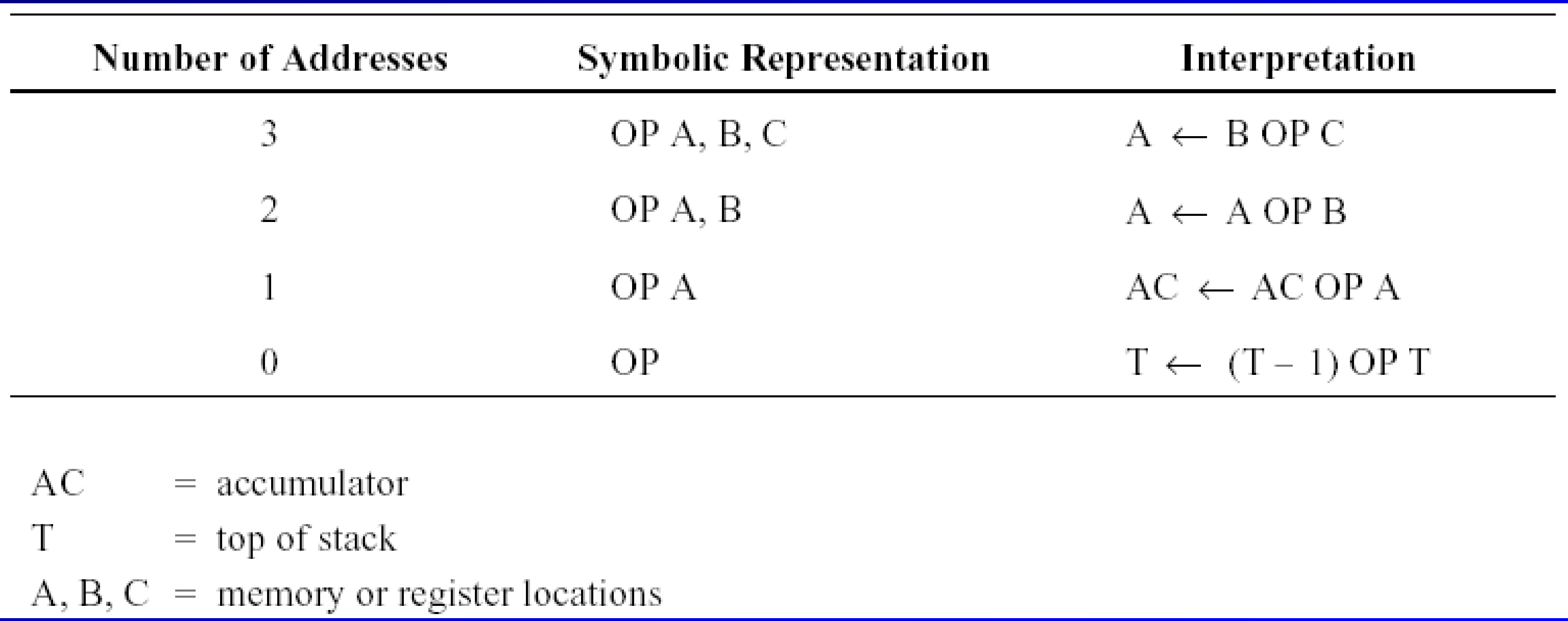
- 1 address
  - Implicit second address
  - Usually a register (accumulator)
  - ADD A
  - AC $\longleftarrow$ AC + A
  - Common on early machines

Number of Addresses

- 0 (zero) address
  - All addresses implicit
  - Uses a stack
  - Compute C = A + B

```
PUSH A
PUSH B
ADD
POP C
```

Number of Addresses Summary



Number of Addresses	Symbolic Representation	Interpretation
3	OP A, B, C	A $\leftarrow$ B OP C
2	OP A, B	A $\leftarrow$ A OP B
1	OP A	AC $\leftarrow$ AC OP A
0	OP	T $\leftarrow$ (T – 1) OP T

AC	=	accumulator
T	=	top of stack
A, B, C	=	memory or register locations

How Many Addresses

- More addresses
  - More complex (powerful) instructions
  - More registers
    - Inter-register operations are quicker
  - Fewer instructions per program
- Fewer addresses
  - Less complex (less powerful) instructions
  - More instructions per program
  - Faster fetch/execution of instructions

Design Decisions

- How many ops?
- What can they do?
- How complex are they?

- Data types
- Instruction formats
  - Length of op code field
  - Number of addresses

Types of Operand

- Addresses
- Numbers
  - Fixed point = Integer
  - Floating point
- Logical Data
  - Bits or flags

Arithmetic

- Add, Subtract, Multiply, Divide
- May include
  - Increment (a++)
  - Decrement (a--)
  - Negate (-a)

Logical

- Shift and Rotate Operations
- Bitwise operations
- AND, OR, NOT

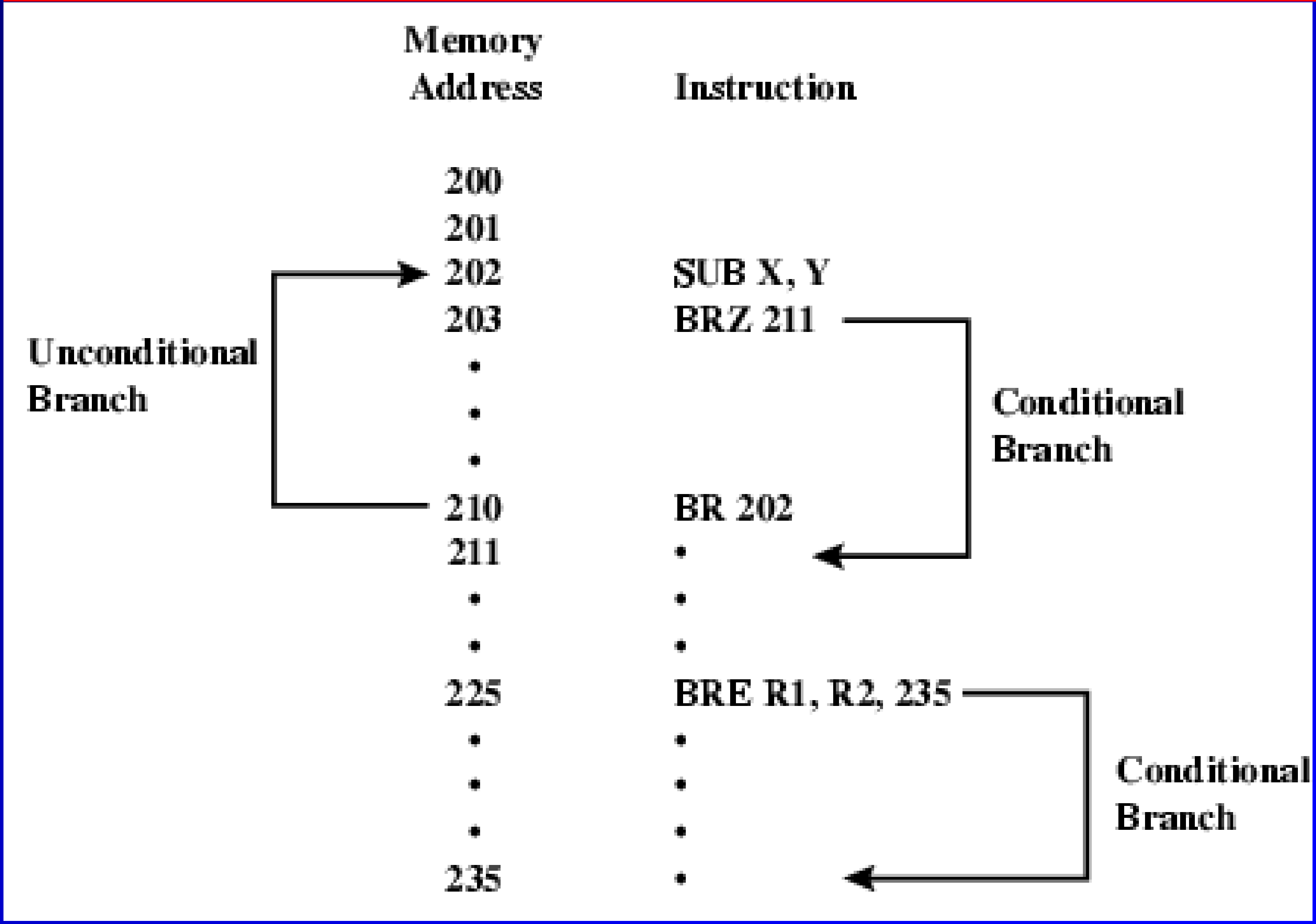
Input/Output

- May be specific instructions
- IN
- OUT

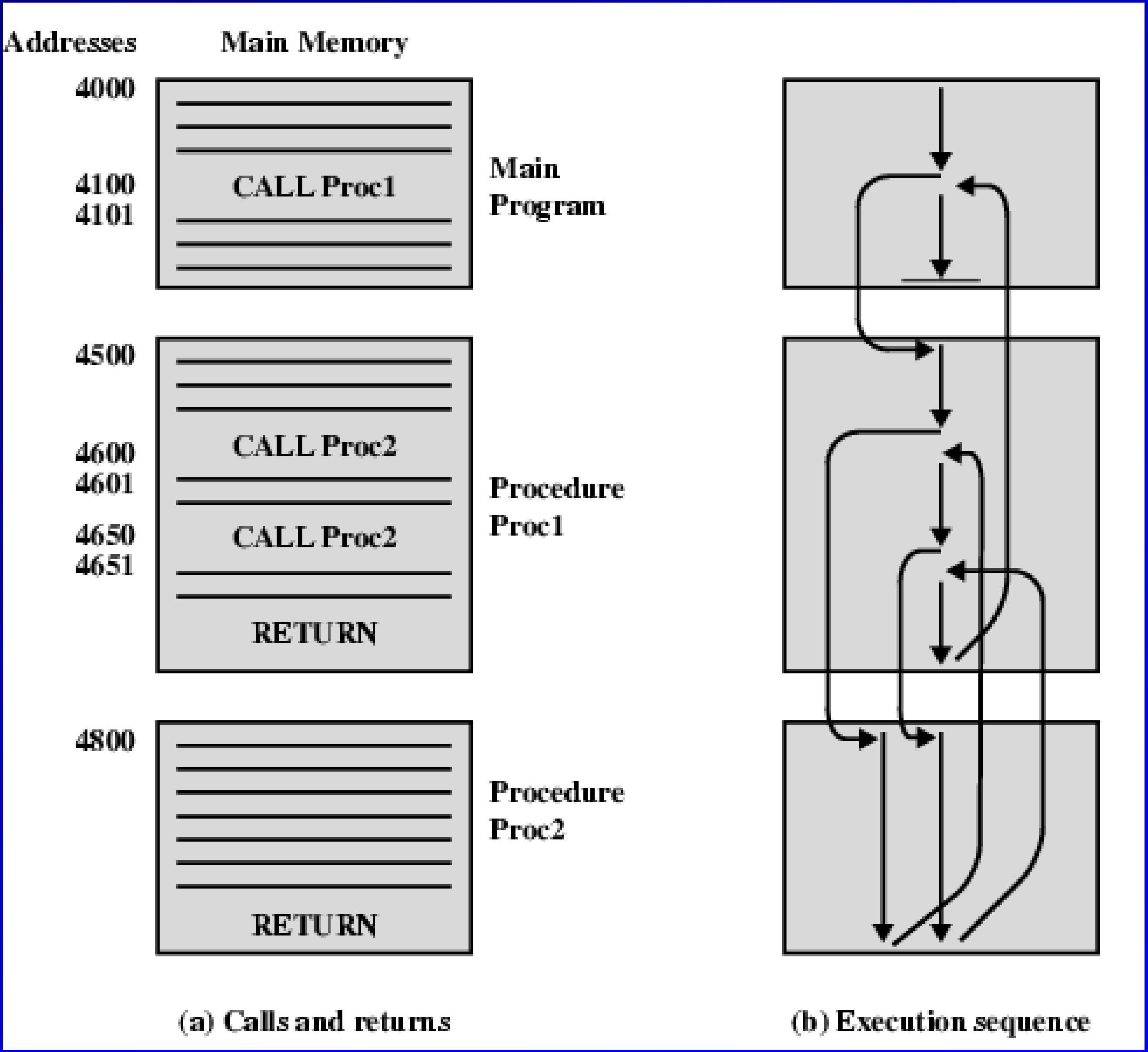
Transfer of Control

- Branch
  - e.g. branch to x if result is zero
- Subroutine call
  - E.g. interrupt call

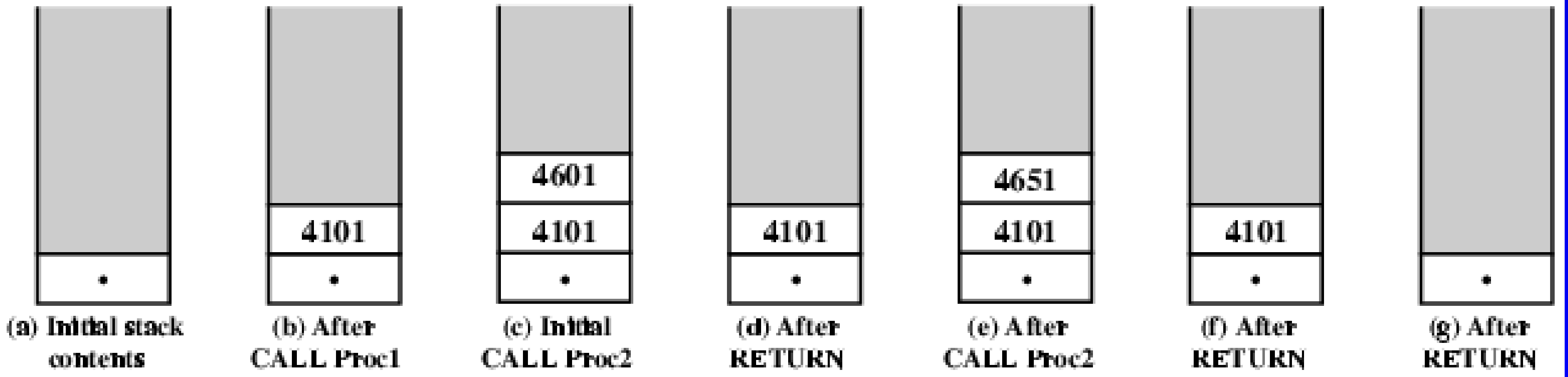
Branch Instruction



Nested Procedure Calls



Use of Stack



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

Byte Order

- What order do we read numbers that occupy more than one byte
- e.g. (numbers in hex to make it easy to read)

Byte Order (example)

12345678 can be stored in memory as follows

Address	Version 1 (big)	Version 2 (little)
0x00184	12	78
0x00185	34	56
0x00186	56	34
0x00187	78	12