Instruction Set Design

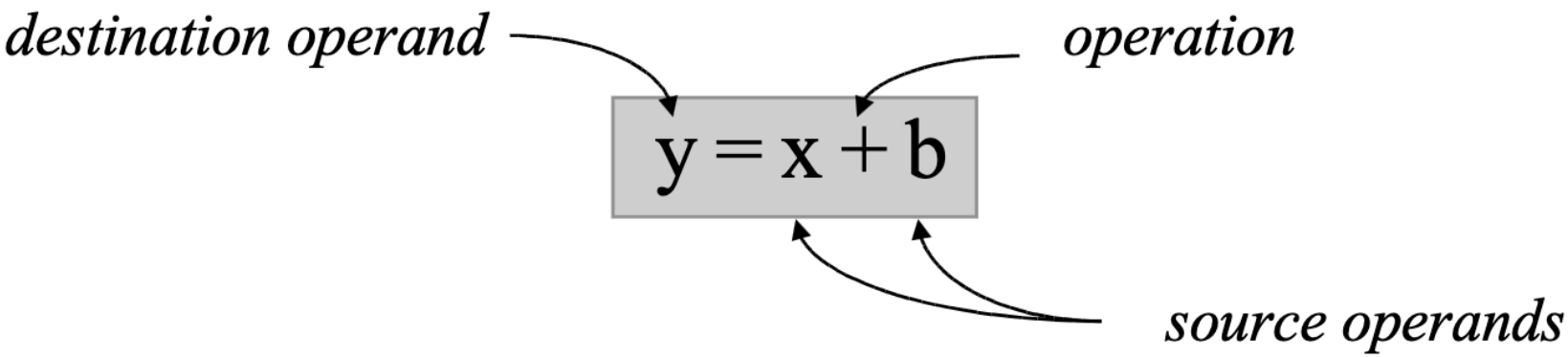
Chapter 12: Operands and Operations

Joannah Nanjekye

July 30, 2024

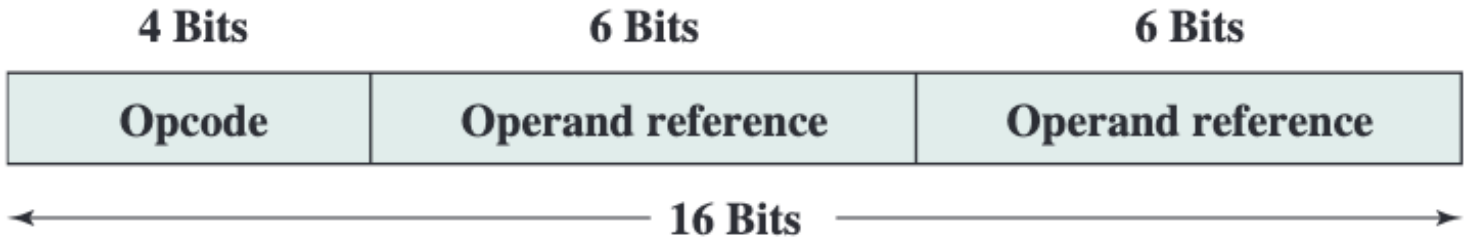
Elements of a Machine Instruction

1. Operation code
2. Source operand reference
3. Result operand reference
4. Next instruction reference



(add r1, r2, r5)

Instruction Representation



- Instructions are represented with a sequence of bits
- The processor extracts the data from instruction fields
- Normally use a symbolic representation of machine instructions
- The opcodes and operand references are converted to to binary form
- Then binary machine instructions are constructed

Instruction Types

- Data processing
  - Arithmetic and Logic instructions
  - Arithmetic instructions process numeric data
  - Logic (boolean) instructions operate on the bits of a word
- Data storage
  - Memory instructions for moving data
  - Between memory and registers
- Data movement
  - I/O instructions
  - Between memory and the user
- Control (test and branch operations)
  - Test instructions test a data word and computation status
  - Branch instructions give control to a different set of instructions

Number of Addresses

$$\text{Programs to Execute Y} = \frac{A - B}{C + (D \times E)}$$

Instruction		Comment
SUB	Y, A, B	Y ← A − B
MPY	T, D, E	T ← D × E
ADD	T, T, C	T ← T + C
DIV	Y, Y, T	Y ← Y ÷ T

(a) Three-address instructions

Instruction		Comment
MOVE	Y, A	Y ← A
SUB	Y, B	Y ← Y − B
MOVE	T, D	T ← D
MPY	T, E	T ← T × E
ADD	T, C	T ← T + C
DIV	Y, T	Y ← Y ÷ T

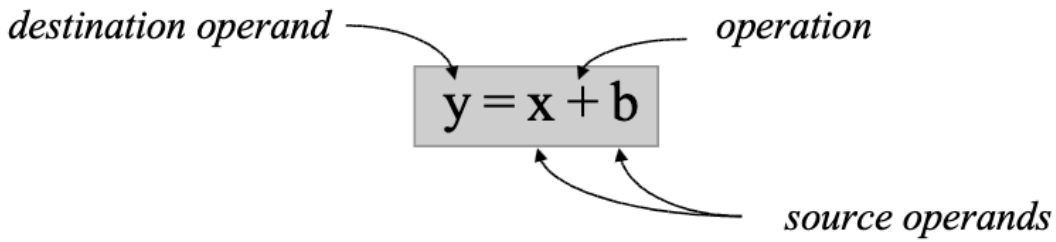
(b) Two-address instructions

Instruction		Comment
LOAD	D	AC ← D
MPY	E	AC ← AC × E
ADD	C	AC ← AC + C
STOR	Y	Y ← AC
LOAD	A	AC ← A
SUB	B	AC ← AC − B
DIV	Y	AC ← AC ÷ Y
STOR	Y	Y ← AC

(c) One-address instructions

Aspects of Instruction Set Design

1. Operation repertoire
2. Data types
3. Instruction format
4. Registers
5. Addressing



(add r1, r2, r5)

- operations
  - how many?
  - which ones
- operands
  - how many?
  - location
  - types
  - how to specify?
- instruction format
  - size
  - how many formats?

how does the computer know what 0001 0100 1101 1111 means?

Operands

Machine instructions operate on data. The most important general categories of data are:

- Addresses
- Numbers
- Characters
- Logical data

Numbers

- Numbers in computers are limited (magnitude and precision)
- Rounding, overflow and underflow have consequences
- Common types of numerical data:
  - Binary integer or binary fixed point
  - Binary floating point
  - Decimal, most common, packed **decimal**

Characters

- Characters are represented with a sequence of bits
- Common codes:
  - International Reference Alphabet (IRA) or
  - American Standard Code for Information Interchange (ASCII) - 7-bit pattern
  - Extended Binary Coded Decimal Interchange Code (EBCDIC) - 8-bit pattern
- All compatible to packed decimal format

Logical Data

- n-bit unit viewed as consisting of n 1-bit items of data
- Each item having the value 0 or 1
- This view of data is considered logical data
- Advantages:
  - Efficient storage for n array of Boolean or binary data items, of either 0 or 1
  - Allow for bit manipulation
- The ”type” of a unit of data is determined by the operation

Operations

Machines support different opecodes that all fall in the following category of operations:

- Data transfer
- Arithmetic
- Logical
- Conversion
- I/O
- System control
- Transfer of control

Data Transfer

- Move data from one location to another
- These instructions must specify the following:
  - The location of the source and destination operands
  - The length of data
  - The mode of addressing for each operand

Data transfer	Move (transfer)	Transfer word or block from source to destination
	Store	Transfer word from processor to memory
	Load (fetch)	Transfer word from memory to processor
	Exchange	Swap contents of source and destination
	Clear (reset)	Transfer word of 0s to destination
	Set	Transfer word of 1s to destination
	Push	Transfer word from source to top of stack
	Pop	Transfer word from top of stack to destination

Arithmetic

- Includes basic arithmetic:
  - Add
  - Subtract
  - Multiply
  - Divide
- For both signed integers and floating point numbers
- Support other unary instructions:
  - Absolute: Take the absolute value of the operand
  - Negate: Negate the operand
  - Increment: Add 1 to the operand
  - Decrement: Subtract 1 from the operand

Logical

Key for bit manipulation and based on boolean operations

- Bitwise logical operations

$$(R1) = 10100101$$
$$(R2) = 11111111$$
$$(R1)\ \text{XOR}\ (R2) = 01011010$$

- Shifting and rotating operations

Input	Operation	Result
10100110	Logical right shift (3 bits)	00010100
10100110	Logical left shift (3 bits)	00110000
10100110	Arithmetic right shift (3 bits)	11110100
10100110	Arithmetic left shift (3 bits)	10110000
10100110	Right rotate (3 bits)	11010100
10100110	Left rotate (3 bits)	00110101

Conversion

- Operate on the format of data
- E.g convert from decimal to binary
- Assume the EAS/390 Translate (TR) instruction
- And we have to translated from EBCDIC to IRA
  - Locations 2100–2103 contain F1 F9 F8 F4.
  - R1 contains 2100.
  - R2 contains 1000.

Then, if we execute

TR R1 (4), R2

locations 2100–2103 will contain 31 39 38 34.

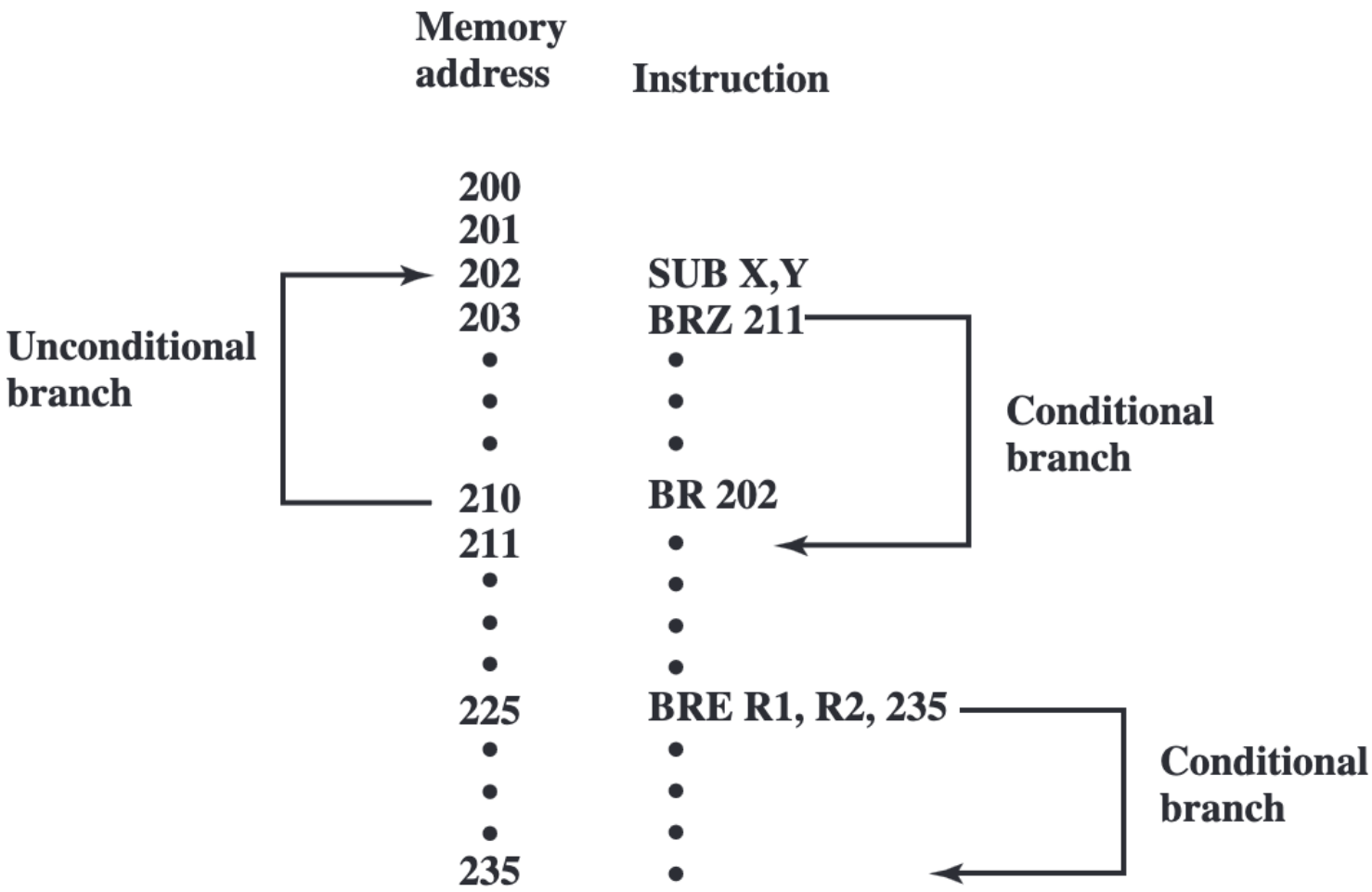
Input/Output and System Control

- Input/output instructions use several approaches:
  - Isolated programmed I/O
  - Memory- mapped programmed I/O
  - DMA
  - The use of an I/O processor.
- System control instructions
  - Executed while the processor is in privileged mode
  - Reserved for the use of the operating system
  - E.g read or alter a control register, read or modify a storage protection key, process control blocks in a multiprogramming system

Transfer of Control

- Supports changing the sequence of instruction execution by updating the PC
- Use cases for transfer-of-control operations:
  - Repeated execution of instructions in loops
  - Programs that involve decision making
  - Modularity
- Common transfer-of-control operations:
  - Branch instructions
  - Skip instructions
  - Procedure call instructions

Branch Instructions

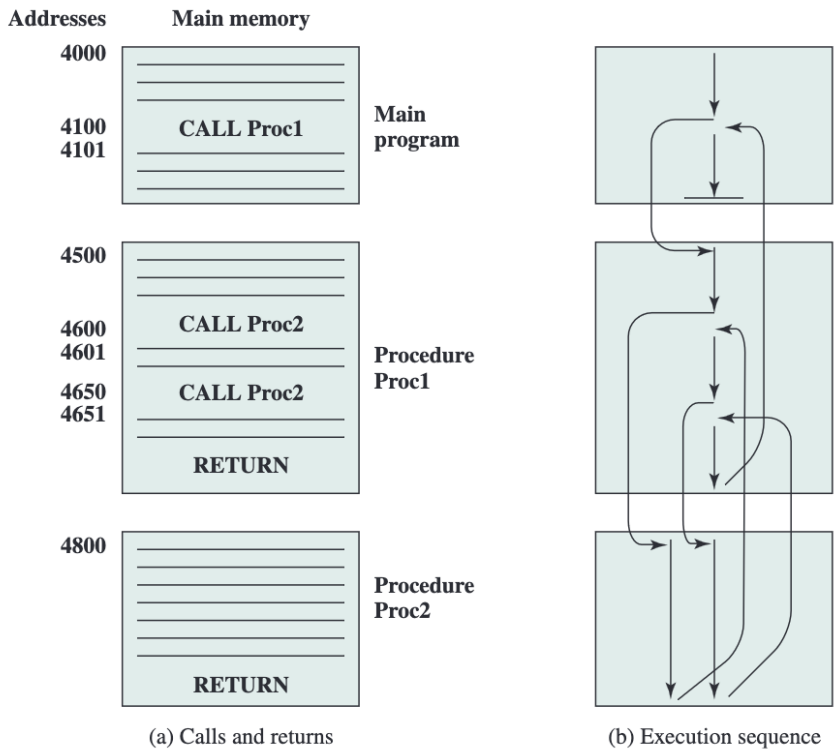


Skip Instructions

```
301
 .
 .
 .
309   ISZ   R1
310   BR    301
311
```

Procedure Call Instructions

- A procedure is a self- contained computer program
- The processor executes the entire procedure and then returns
- Procedures are used for economy and modularity



Saving a return Address

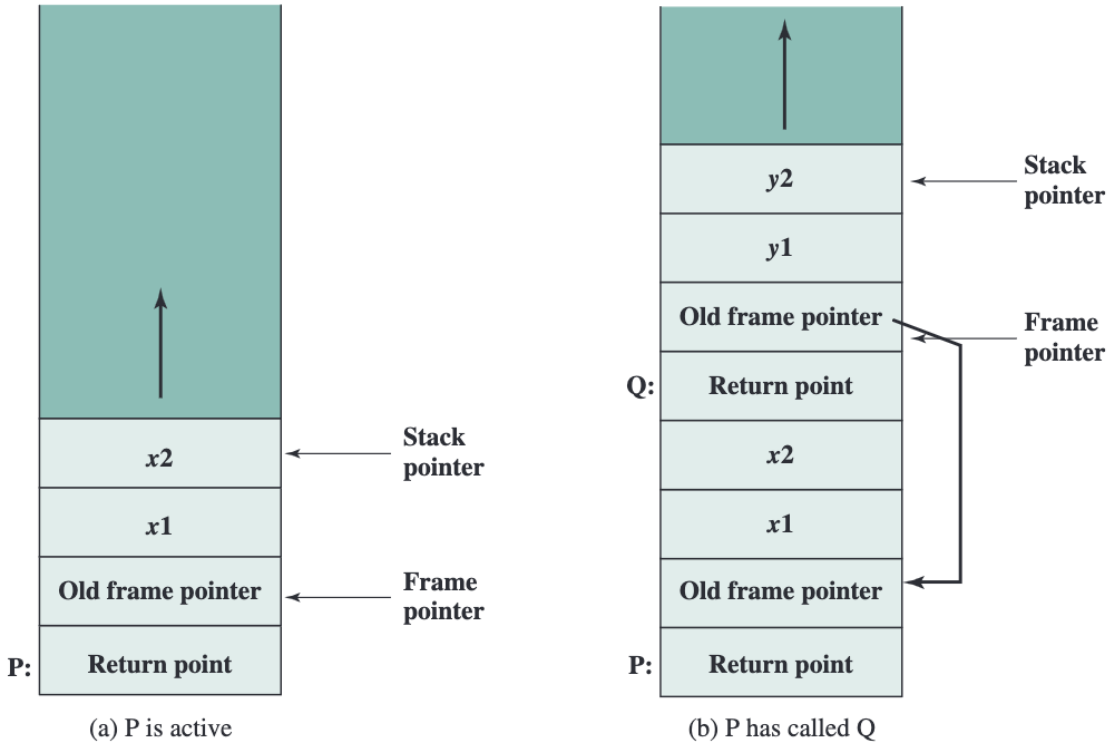
Three ways of saving state

- Register
- Start of called procedure
- Stack

$$RN \leftarrow PC + \Delta$$
$$PC \leftarrow X$$

$$X \leftarrow PC + \Delta$$
$$PC \leftarrow X + 1$$

Stack: Return and Parameter Addresses



Acknowledgement of Sources

- Most of these slides are adopted from Dr. Nader Taleb lectures
- https://slideplayer.com/slide/10852334/
- https://cseweb.ucsd.edu/classes/su06/cse141/slides/s02-isa-1up.pdf
- https://slideplayer.com/slide/9163053/
- https://h3turing.vmhost.psu.edu/cmpsc312/ExpOpcodes1.pdf