# Instruction Sets: Addressing Mode and Formats

Computer Organization and Architecture

SOUTH CHINA UNIVERSITY OF TECHNOLOGY
FALL 2016
DR. MAO Aihua
ahmao@scut.edu.cn

## Instruction Mode and Formats

- In Ch.10, we focused on what can an instruction do, specially the types of operands and operation
- This Chapter, turns to how to specify the operands and operation of instruction
  - *How the address of operand specified*
  - *How the bits of an instruction are organized (operand address, opcode)*

2

## Instruction Mode and Formats

- The address fields in instruction format actually is *small*
- But we would like to address *more location* in the main memory or virtual memory
- To achieve this, a variety of addressing techniques has been employed
- Involve some trade off between the *address range and addressing complexity*

3

## Addressing Modes

- Immediate
- Direct
- Indirect
- Register
- Register Indirect
- Displacement (Indexed)
- Stack

4

## Two Comments

- How the processor determine which address mode is being used for a particular instruction?
  - Different opcode will use different addressing mode
  - One or more bits in a instruction is used as mode fields to indicate the addressing mode

5

## Two Comments

- Interpretation of the effective address
  - Is either main memory address or register without virtual memory
  - The actual mapping to physical address *is a function of the paging mechanism*

6

## Immediate Addressing

- Simplest, *Operand value* is part of instruction
- Operand = A
- e.g. ADD 5
- No memory reference to fetch data, *saving one memory or cache cycle*
- Fast
- Limited range, the size of number is restricted to the size of the address field

7

## Immediate Addressing Diagram


ADD 5

: Add 5 to contents of accumulator

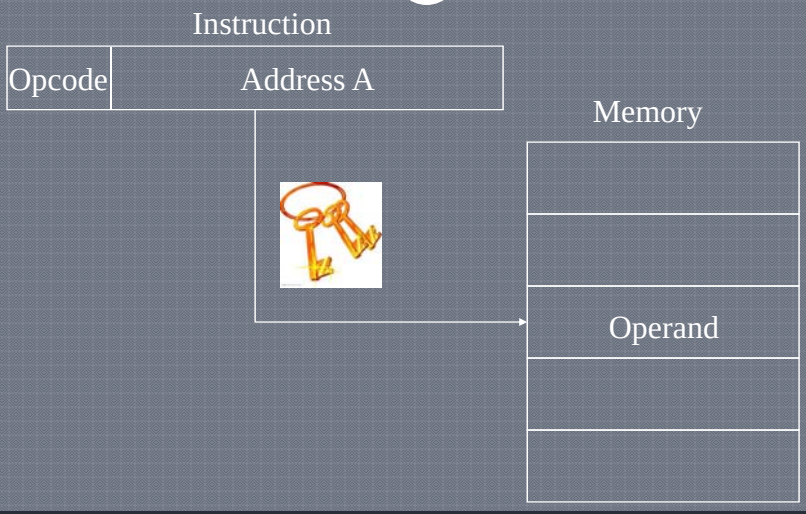
8

## Direct Addressing

- Address field contains *effective address of operand*
- Effective address (EA) = address field (A)
- e.g. ADD A
  - Add contents of cell A to accumulator
  - Look in memory at address A for operand

9

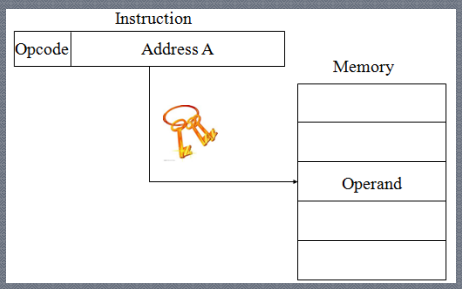
## Direct Addressing Diagram




10

## Direct Addressing

- Single memory reference to access data
- No additional calculations to work out effective address
- Limited address space, address field usually less than the word length， why?



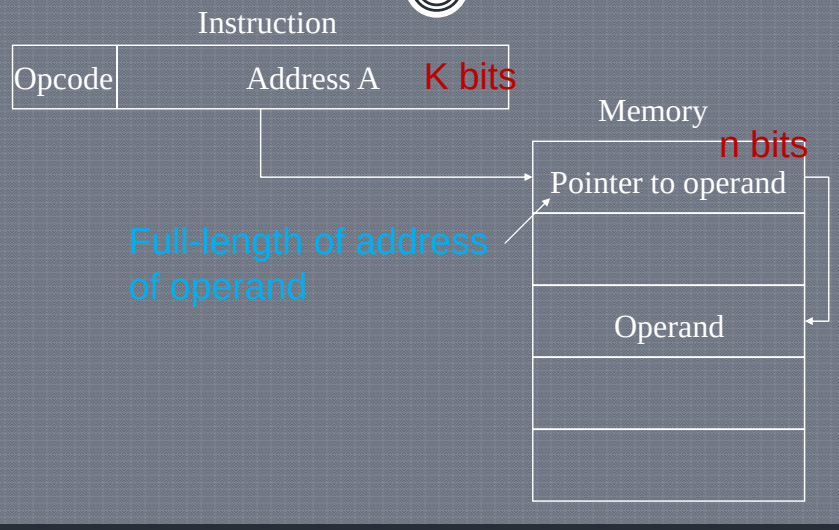

11

## Indirect Addressing

- The address field refer to *the address of a word* in memory, which contains a *full-length address of the operand*
- EA = (A)
  - Parentheses means the contents inside
- e.g. ADD (A)
  - Add contents of cell pointed by contents of A to accumulator

12

## Indirect Addressing Diagram




13

## Indirect Addressing

- Large address space
- n = word length, Address space $2^n$ is available
- Disadvantage is need two memory reference to fetch the operand
- Though the addressable space is $2^n$, the number of different effective address at one time is $2^k$ (k is the length of address field)
- An indirect memory reference may involve at least one page fault (n>k)

14

## Indirect Addressing

- May be nested, multilevel, cascaded
  - e.g. EA = (((A)))
    - Draw the diagram yourself
    - What is the new features?

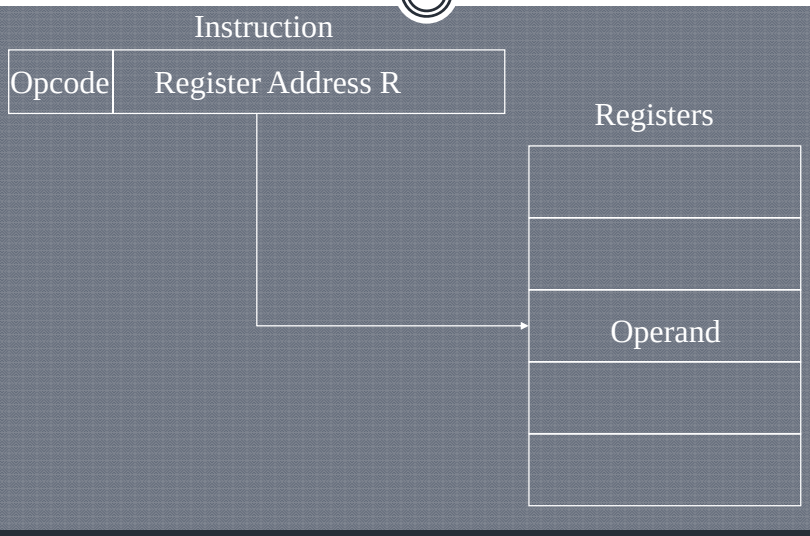
15

## Register Addressing

- The address field is *a register*
- Operand is held in this register
- EA = R
- If the content of register address field R is 5, means R5 is the intended address
- Limited number of registers, if the address filed have 3~5bits, then total 8~32 registers can be referred

16

## Register Addressing Diagram




17

## Register Addressing

- Advantages:
  - Very small address field needed
    - Shorter instructions
    - Faster instruction fetch
  - No time-consuming memory access
  - Very fast execution
- Disadvantages:
  - Very limited address space

18

## Register Addressing

- If register address is heavily used, means *the processor registers will be heavily used*
- Due to the *limited number of registers*, it is wise to keep the operand in a register *remains in use for multiple operations*, avoid the wasteful intermediate step

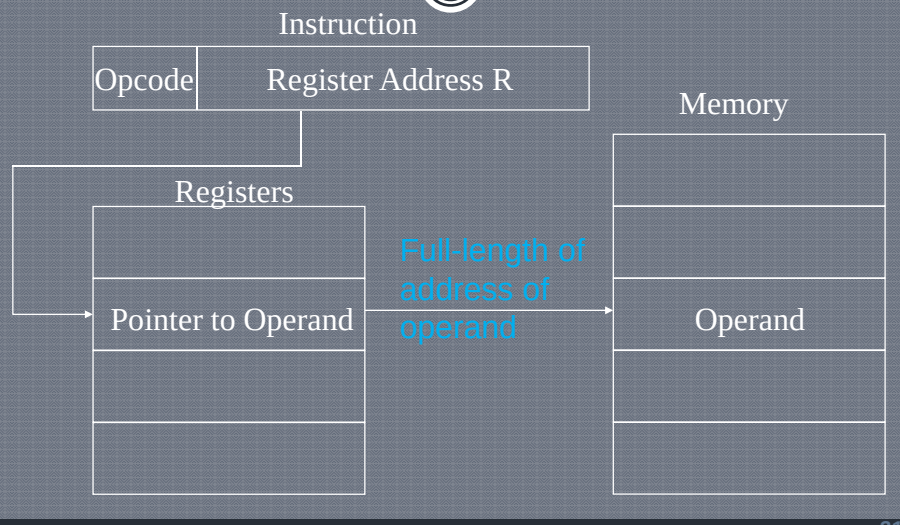
19

## Register Indirect Addressing

- Is analogous to indirect addressing, difference is the address filed refers to a register
- EA = (R)
- Advantages and disadvantages are same to indirect addressing
- Use one less memory access than indirect addressing

20

## Register Indirect Addressing Diagram



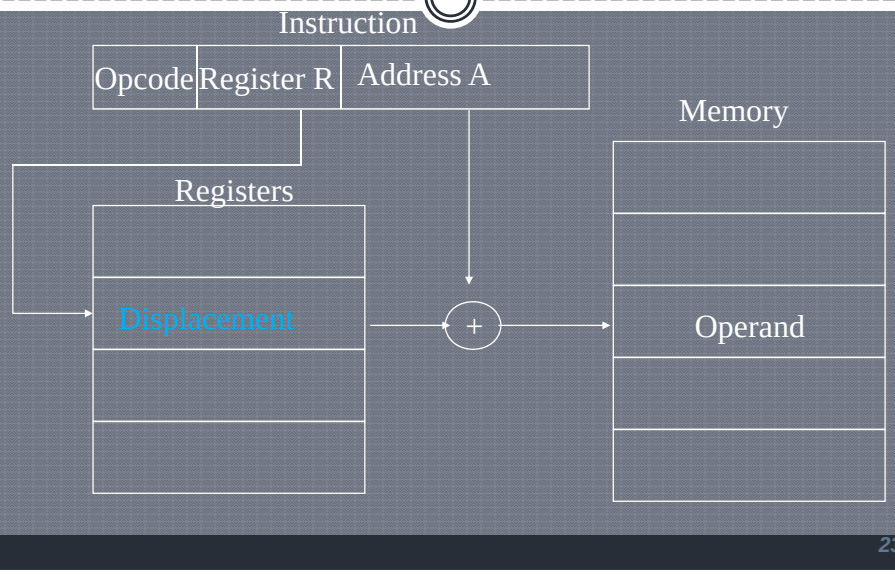

21

## Displacement Addressing

- Combines the capabilities of direct addressing and register indirect addressing
- EA = A + (R)
- Address field hold two values
  - *A = base value, used directly*
  - *R = register that holds displacement , added to A to produce the effective address*

22

## Displacement Addressing Diagram




23

## Displacement Addressing

- Most common use of displacement addressing
  - Relative addressing
  - Base-register addressing
  - Indexing

24

## Relative (PC-relative) Addressing

- A version of displacement addressing
- EA = A + (R)
- R = Program counter, PC
- EA = A + (PC)
- i.e. get operand from *A added with the next instruction address* pointed by PC
- Exploit the *concept of locality of reference*

25

## Base-Register Addressing

- EA = A + (R) → Is base, variable
- R holds a main memory address
- A holds displacement (an unsigned integer) from that address
- R may be explicit or implicit
- e.g. segment registers in 80x86

26

## Indexed Addressing

- EA = A + (R)
- A = base, a main memory address
- R = a *positive displacement* from that address
- Used for performing iterative operation
  - A, A+1, A+2, A+3, A is the start location of the list, R is index register
- Autoindexing, do as a part of the same instruction cycle
  - EA = A + R
  - R++

27

## Combinations

- Both *indirect addressing and indexing* are provided in the same instruction
- If index performed after indirection, termed Postindex
- EA = (A) + (R)
- Such as the process control block by OS, index register contains the displacement with the block

28

## Combinations

- Indexing performed before indirection, Preindex
- EA = (A+(R))
- Multiway branch table, may be a branch to a number of point from a particular point in a program depended on conditions

29

## Stack Addressing

- Operand is (implicitly) on top of stack
- e.g.
  - ADD Pop top two items from stack
- The stack point is maintained in a register, so it is in fact register indirect address

30

## 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
- The most basic design issue is *the instruction format length*

31

## Instruction Length

- Affected by and affects:
  - *Memory size*
  - *Memory organization*
  - *Bus structure*
  - *CPU complexity*
  - *CPU speed*

32

## Instruction Length

- Trade off between powerful instruction repertoire and saving space
- More opcodes and operand is possible to write short program and more flexibility, but longer instruction may be wasteful
- Instruction length should be equal to the memory-transfer length.
- Shorter instruction is a solution to the bottleneck of memory transfer rate

33

## Allocation of Bits

- A trade-off between the number of opcodes and the power of addressing capability
- More bits in the opcode means few bits for addressing
- One interested refinement is use of variable-length opcode
  - *A minimum code length*, additional operations be specified using additional bits

34

## Allocation of Bits

- Factors to determine the addressing bits
  - Number of addressing modes:
    - one or more bits are needed to indicate the addressing mode
  - Number of operands:
    - fewer address make for longer, awkward program
  - Register versus memory:
    - only a few bits are needed to specify the register, the more registers can be used for operand reference, the fewer bits are needed

35

## Allocation of Bits

- Factors to determine the addressing bits
  - Number of register sets:
    - for a fixed number of registers, a functional split requires fewer bits, such as, with 2 sets of 8 registers, only 3bit are required to identify a register
  - Address range:
    - related to the number of address bits
  - Address granularity:
    - In a system with 16 or 32 bit word, a address can reference a word or byte

36

## Homework

- Review question: 11.3
- Hand in 10th Nov.

37