Chapter2 Instruction System

- Data Type and Data Representation
- Instruction System Design Principle and Optimization
- RISC Computer

Chapter2 Instruction System

Instruction System Design Requirement

1. Comprehensively consider some factors: computer application, hardware organization, OS, etc.
2. Software portability: Make the software backward compatibility, and strive to upward compatibility.

Chapter2 Instruction System

Instruction System Design Rules

- **Completeness**
- **Regularity**
- **Uniformity**
- **Orthogonality**
- **Composability**
- **Compatibility**
- **Scalability**

Chapter2 Instruction System

Data Type and Data Representation

Data Type: **A set of data values, but also defines the action sets can be applied to the collection.**

Classification:

Basic Data Type: bit, integer, etc.
Structured Data Type: array, stack, queue, etc.
Abstract Data Type
Pointer

Chapter2 Instruction System

Data Representation

Data types which can be identified directly by the hardware and can be directly used by instruction system.

Data representation is essentially a trade-off of software and hardware.

Chapter2 Instruction System

Custom Data Representation

In order to shorten the semantic gap when the same data attributes are explained by the machine language and high-level language.

Custom Data Representation: Indicates data type by the data itself, make the data inside the computer has a custom ability.

Classification:

- **Data Representation with Identifier**
- **Data Descriptor**

Chapter2 Instruction System

Data Representation with Identifier

Describe the simple data, identifier is connected to each data value, exists within the same storage unit.

Identifier	Data Value

Chapter2 Instruction System

Data Descriptor

Describe the complex and multidimensional data, such as vector, arrays, records, etc.

Describe the property of data to access.

Descriptor and data word stored separately.

Computer must use descriptor to access each data's address and other information.

Chapter2 Instruction System

Data

000	Data Value

Data Descriptor

3	1	1	1	1	1	2	1	20	20
101	P	C	I	S	R	T	D	Length	Address

D: 0: Single Precision; 1: Double Precision

T: 00: Data Descriptor

R: Read-only

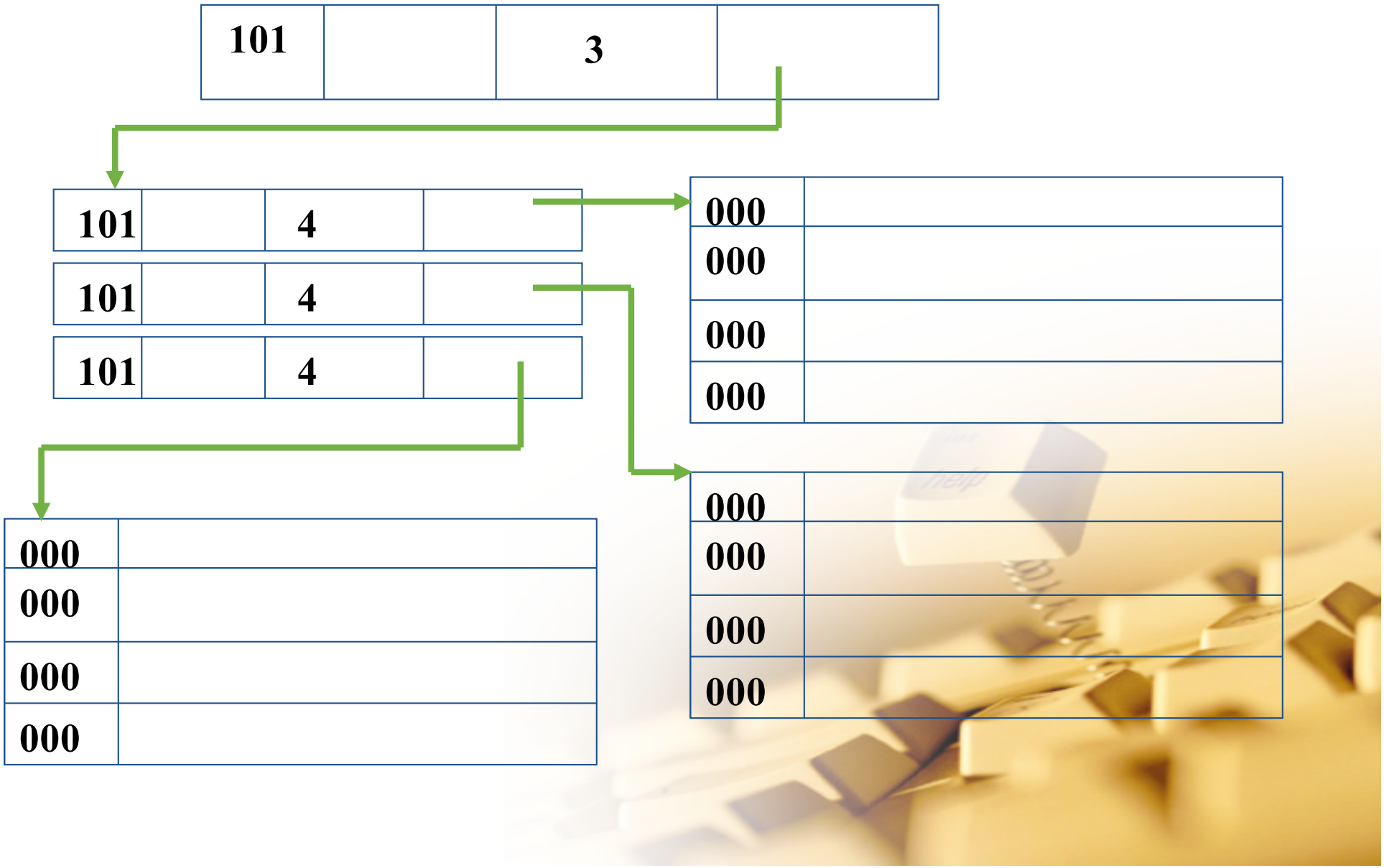
S: 0: Continuous; 1: Non-continuous

I: 0: All data set; 1: One of data set

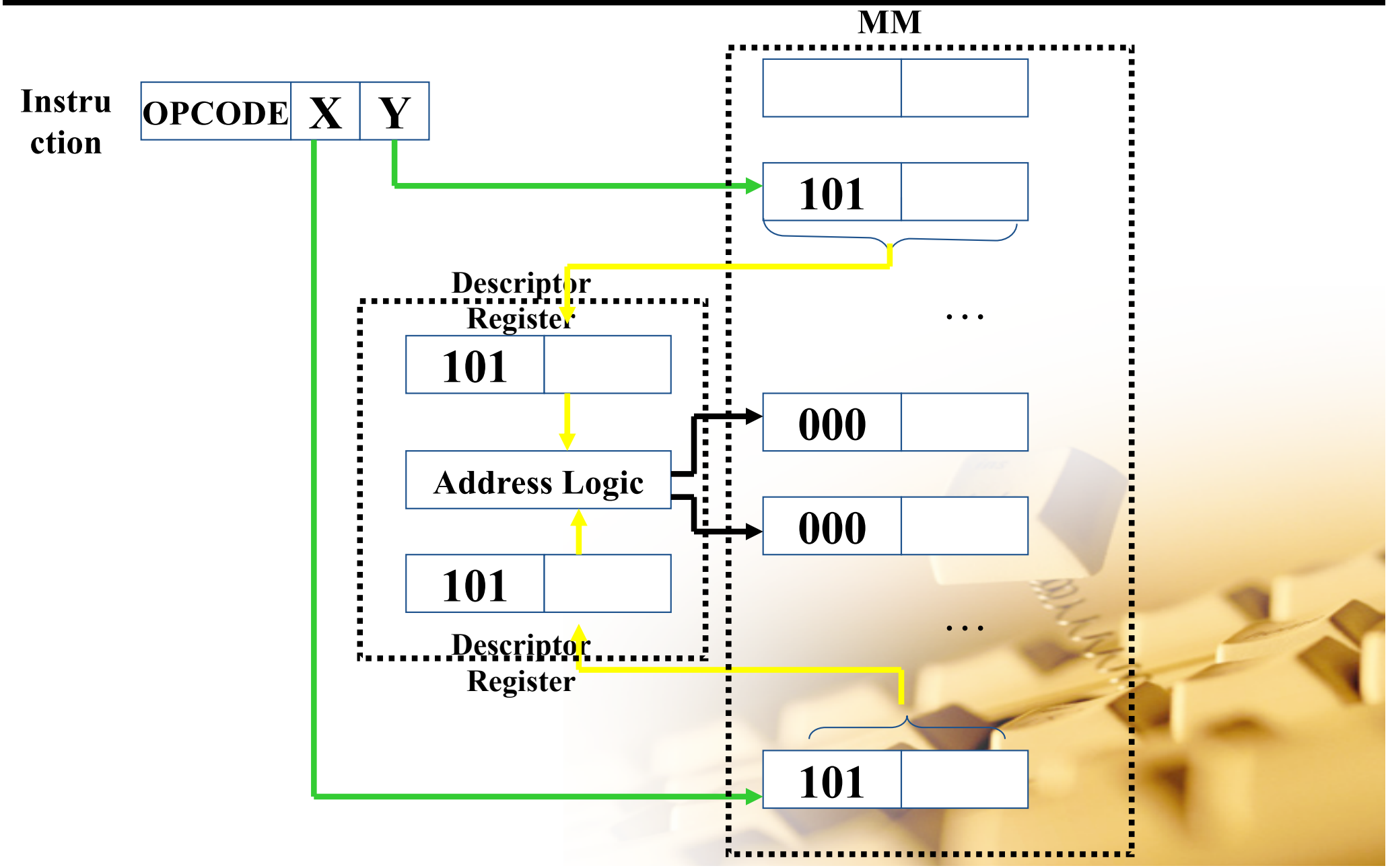
C: Character Data

P: 0: Not in the main memory; 1: In the main memory

Chapter2 Instruction System



Chapter2 Instruction System



Chapter2 Instruction System

Instruction System Design Principle

Instruction System Encoding Method

- Orthogonal method
- Integrated method
- Hybrid method

Chapter2 Instruction System

Instruction System Design Principle

Basic idea:

Some basic operation of the computer system (including the operating system and the operation of the high-level language) should be implemented by software or hardware.

Some complex operation should be implemented by an instruction or by a series of instructions.

Chapter2 Instruction System

◆ Addressing Technology

1. Access Mode

- Access by Address：Serial
- Access by Content：Parallel

2. Program Locate Mode

The process of translating the logical address (the relative address) of instructions and data into main memory physical addresses (the absolute address).

- Direct Locate Mode
- Static Locate Mode
- Dynamic Locate Mode

Chapter2 Instruction System

Instruction Format Optimization

How to use the shortest possible digits to represent instruction operation information and address information, use the shortest possible time to deal with high frequency instruction. How to reduce redundant information in the instruction word and use the least bits of information to represent the required operating information and address information.

Operation Code Optimization

Basic idea of Huffman Compression

When the probability of events are not equal, using optimization technology to encode the high probability events with the shortest possible digits (time) to represent (processing), and to encode the low probability with a longer digits (time) to represent (processing)

Chapter2 Instruction System

Instruction	Probability
I1	0.40
I2	0.30
I3	0.15
I4	0.05
I5	0.04
I6	0.03
I7	0.03

Information Entropy

$$H=-\sum P_i \log_2 P_i$$

Average Coding Length

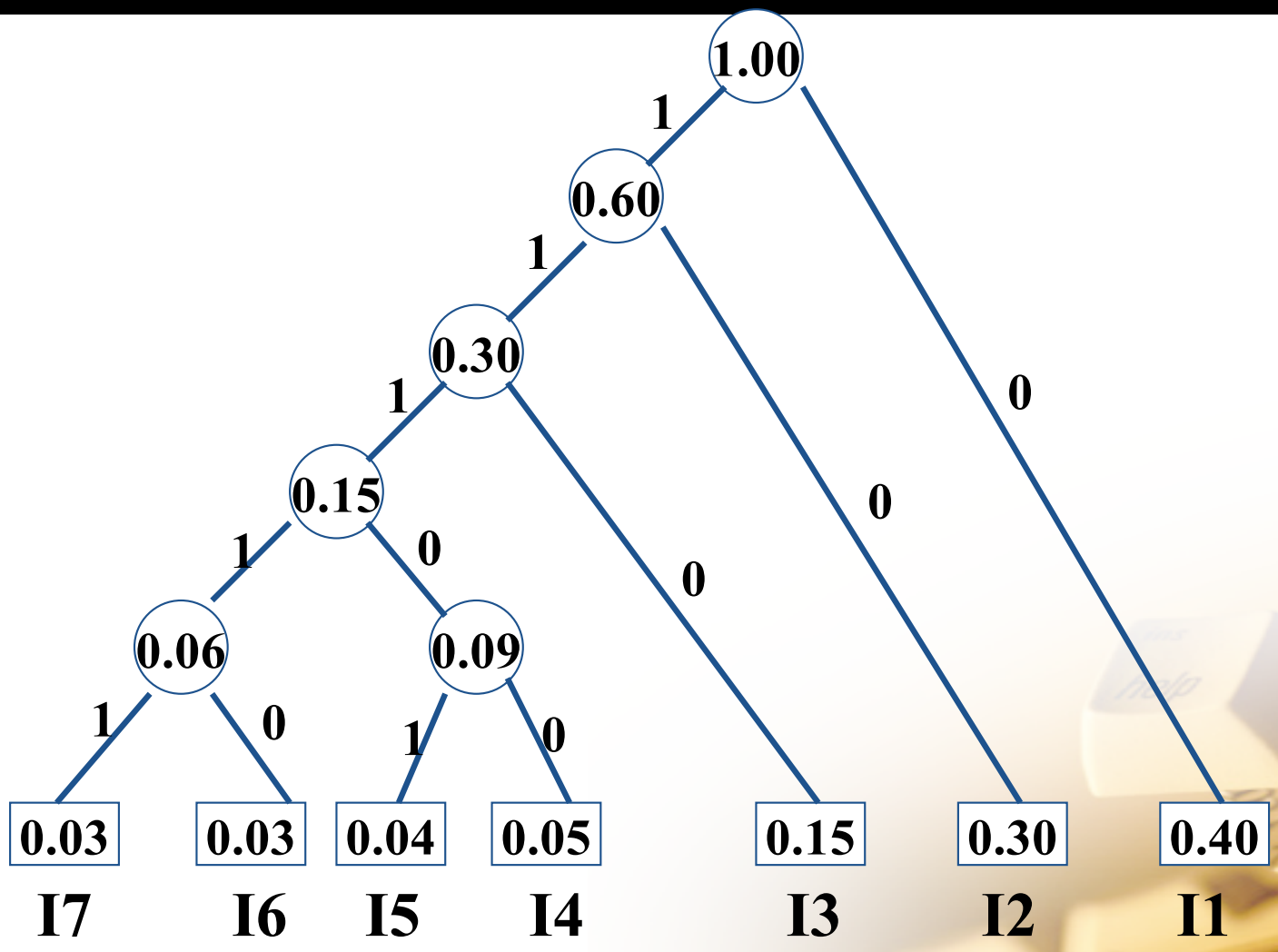
$$L=\sum l_i * P_i$$

$H=-\sum P_i \log_2 P_i=0.40*1.32+0.30*1.74+0.15*2.74+0.05*4.32+0.04$

$*4.64+0.03*5.06+0.03*5.06=2.17$

Information redundancy =1-2.17/3=0.28

Chapter2 Instruction System



$\sum P_i l_i = 0.40*1+0.30*2+0.15*3+0.05*5+0.04*5+0.03*5$
$+0.03*5=2.20$

Information redundancy $= 1-2.17/2.20 \approx 1.36\%$

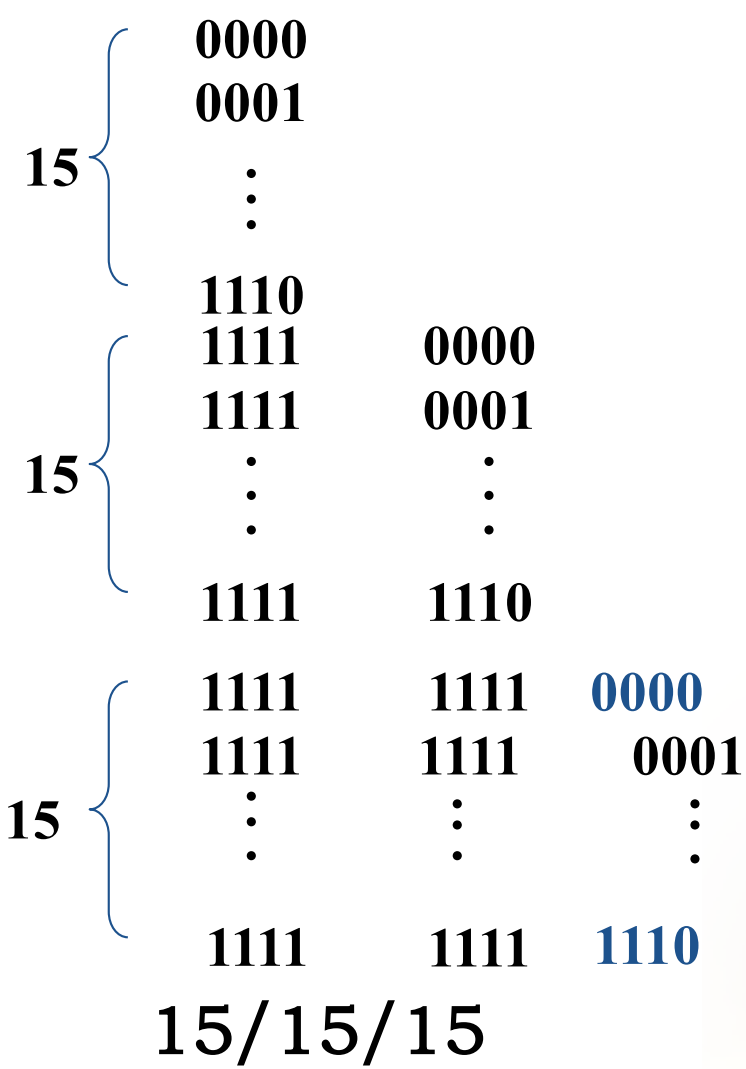
Chapter2 Instruction System

Extension Coding

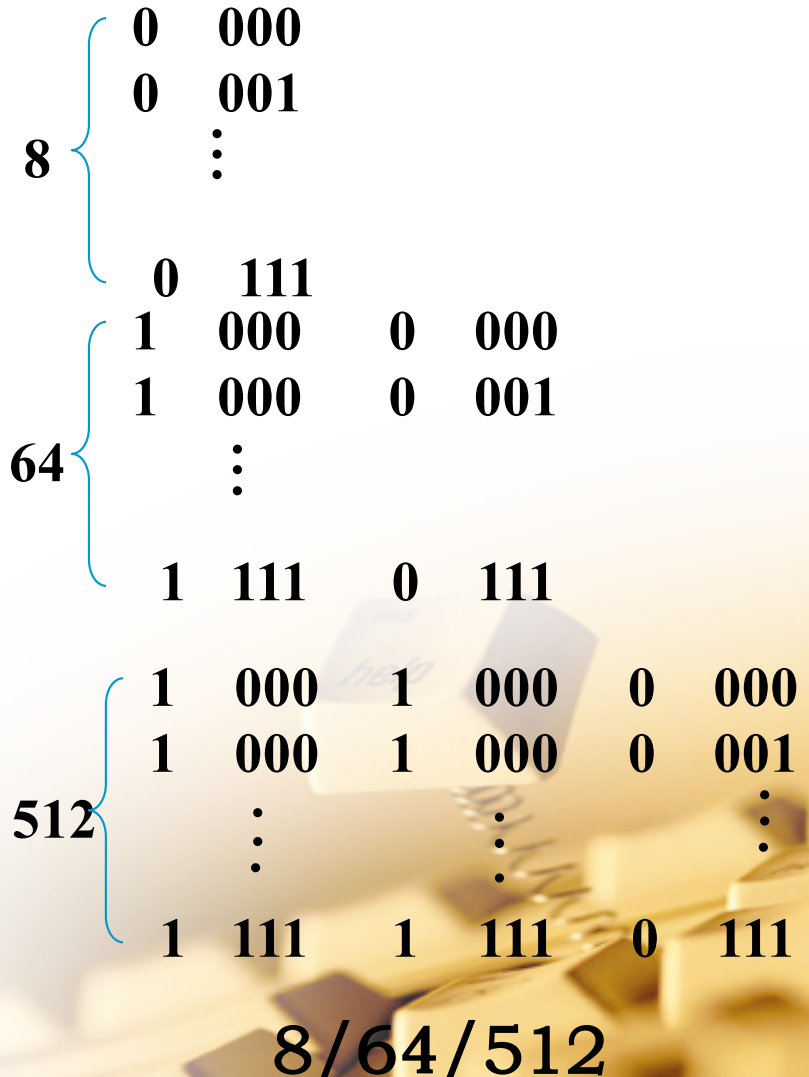
Instruction	Probability	Huffman Coding	Code Length	Huffman Extension Coding	Code Length
I1	0.40	0	1	0 0	2
I2	0.30	1 0	2	0 1	2
I3	0.15	1 1 0	3	1 0	2
I4	0.05	1 1 1 0 0	5	1 1 0 0	4
I5	0.04	1 1 1 0 1	5	1 1 0 1	4
I6	0.03	1 1 1 1 0	5	1 1 1 0	4
I7	0.03	1 1 1 1 1	5	1 1 1 1	4

Average Coding Length=2.30， **Information redundancy** =5.65%。

Chapter2 Instruction System



15/15/15



8/64/512

Chapter2 Instruction System

◆ RISC Computer

The Design Principle of RISC

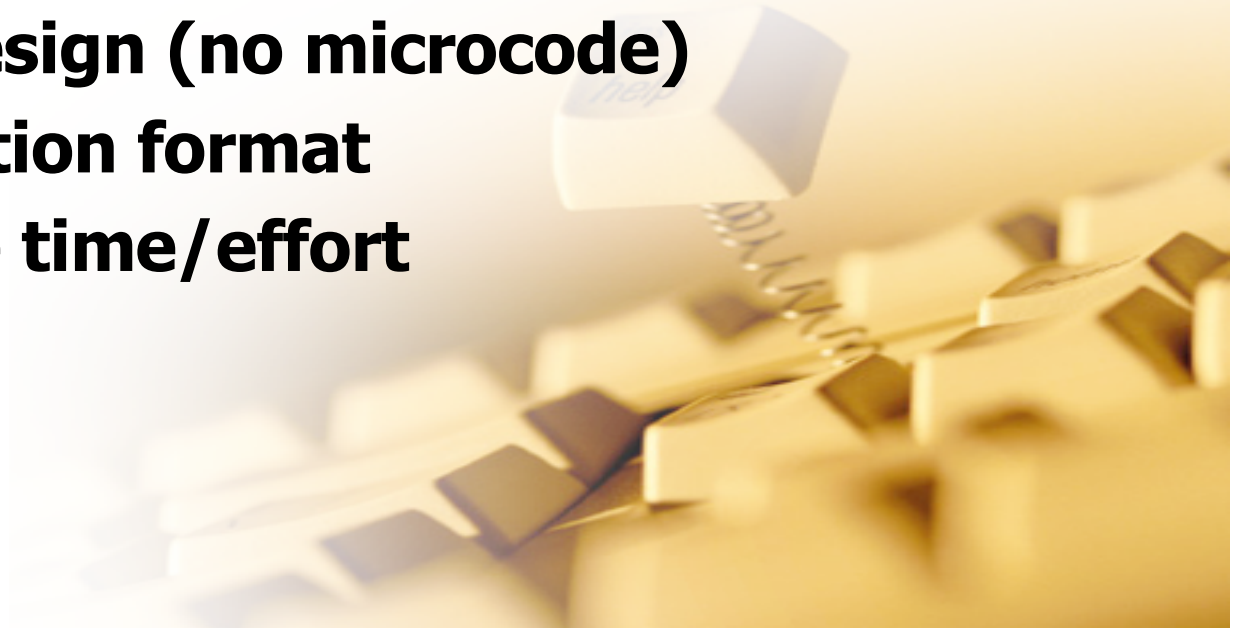
The Origin of RISC

- ❖ 20%-80% Law
- ❖ The Tradeoffs of System Design between Hardware and Software
- ❖ The Development of VLSI

Chapter2 Instruction System

RISC Characteristics

- One instruction per cycle
- Register to register operations
- Few, simple addressing modes
- Few, simple instruction formats
- Hardwired design (no microcode)
- Fixed instruction format
- More compile time/effort



Chapter2 Instruction System

CPI for RISC and CISC

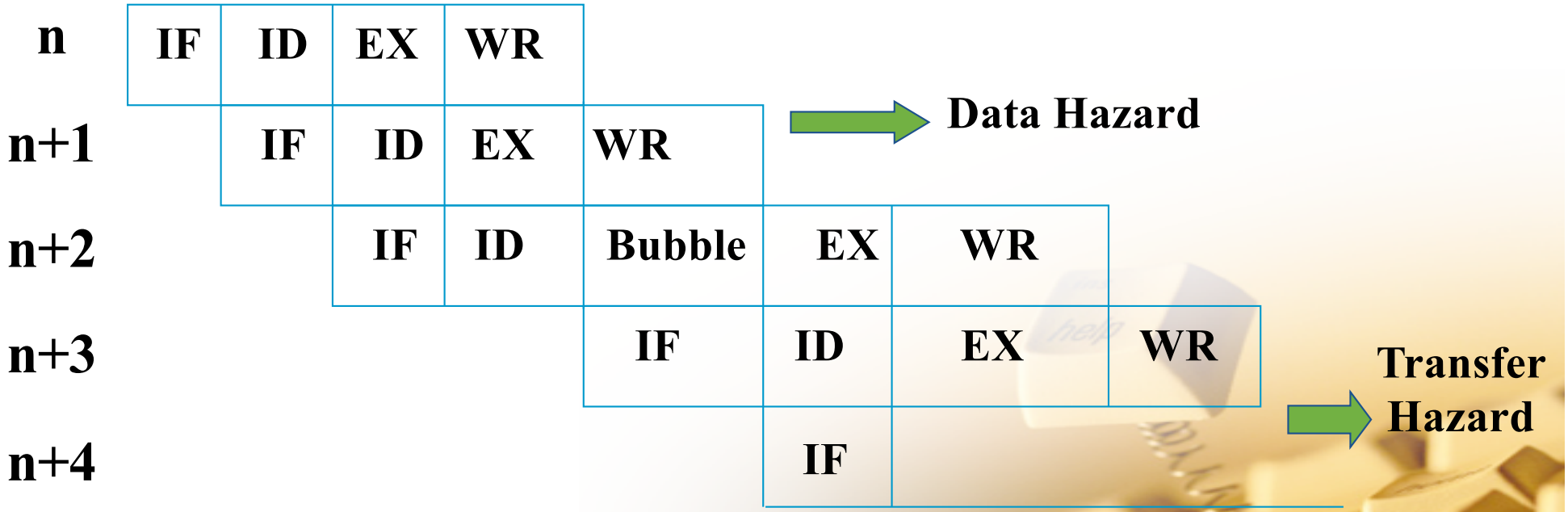
- $T_{cpu}=I_N*CPI*T_c$

Type	I_N	CPI	T_c
CISC	1	2~15	33(ns)~5(ns)
RISC	1.3~1.4	1.1~1.4	10(ns)~2(ns)

Chapter2 Instruction System

The Main Technology of RISC

1. Pipeline Structure and Instruction Scheduling



Chapter2 Instruction System

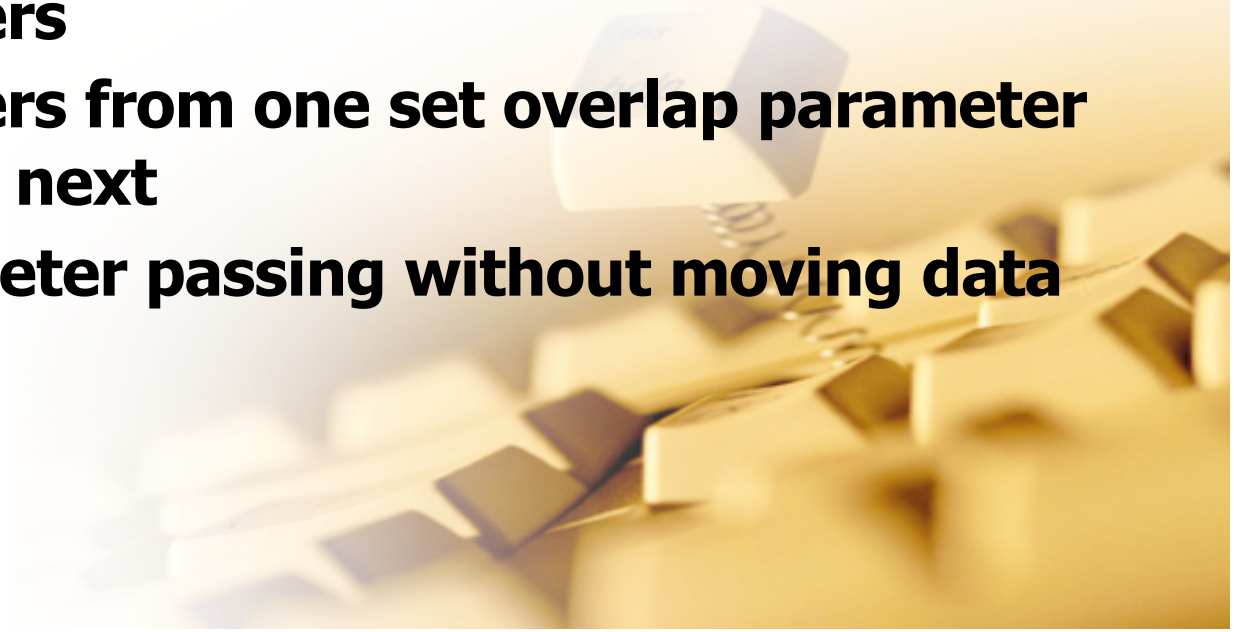
2. Overlapping Register Windows

- Only few parameters
- Limited range of depth of call
- Use multiple small sets of registers
- Calls switch to a different set of registers
- Returns switch back to a previously used set of registers

Chapter2 Instruction System

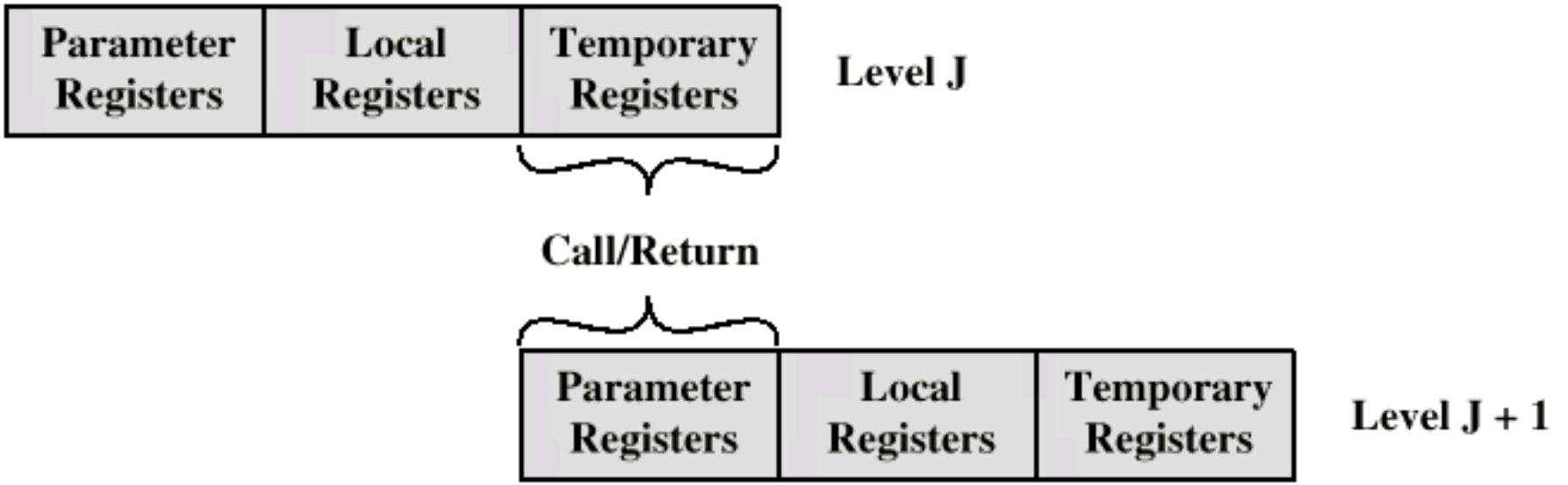
2. Overlapping Register Windows

- Three areas within a register set
  - Parameter registers
  - Local registers
  - Temporary registers
  - Temporary registers from one set overlap parameter registers from the next
  - This allows parameter passing without moving data



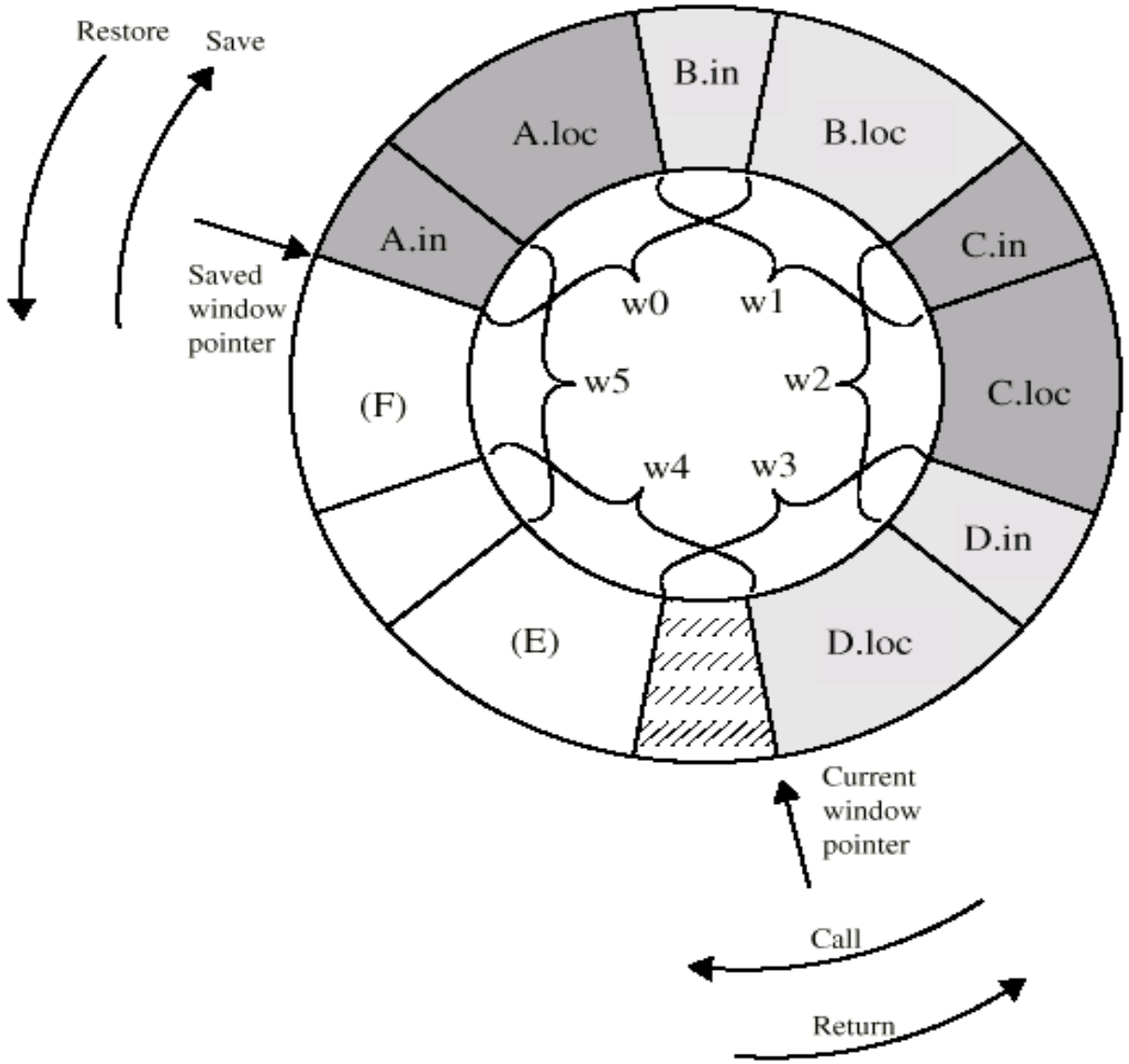
Chapter2 Instruction System

2. Overlapping Register Windows



Chapter2 Instruction System

Circular Buffer diagram



Chapter2 Instruction System

Operation of Circular Buffer

- When a call is made, a current window pointer is moved to show the currently active register window
- If all windows are in use, an interrupt is generated and the oldest window (the one furthest back in the call nesting) is saved to memory
- A saved window pointer indicates where the next saved windows should restore to

Chapter2 Instruction System

3. Compiler Optimization Technology

- How to best allocate registers in the register heap, effectively reduce access to memory data
- Try to reorder and scheduling instructions in the program on the basis of keeping the original semantic and to develop parallelism.