Scalable CPU Architecture

The System-On-Chip Solution

Presented to
NEC Investment Group
By
Dr. C. H. Ting
eForth Technology, Inc.

March 19, 2003

Summary

- Architectural Wastes of CISC and RISC Designs
- The eForth Approach
- eP Series Block Diagrams
- eP Series Instruction Set
- eForth Operating System
- 5 Demonstrations
- Concluding Remarks

● ● ● ● ● ● ● ● ● ● ●
● ● ● ● ● ● ● ● ● ●
● ● ● ● ● ● ● ● ● ●

(• • • • • • •

For knowledge, add a little everyday.
For wisdom, delete a little everyday.
Delete until there is nothing to delete,
And then you can accomplish everything.
Lao-Tze, Tao Te Ching

Architectural Wastes of CISC and RISC Designs

Architectural Wastes of CISC

- Too many instructions
- Too many instruction types
- Too many memory addressing modes
- Some instructions take long times to execute
- Difficult for compiler optimizing
- Difficult for operating system development

Architectural Wastes of RISC

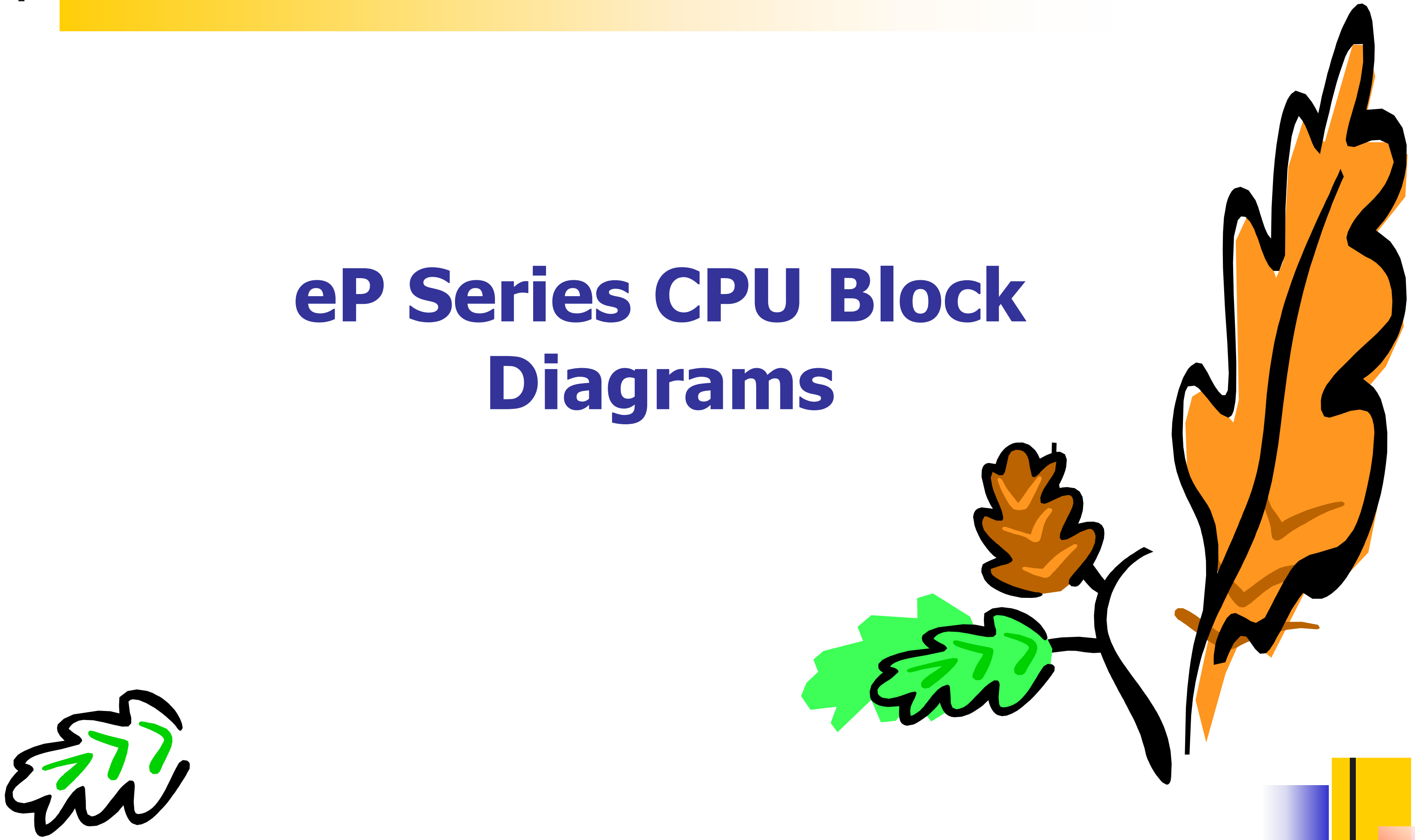
- Inflexible 32-bit architecture
- Too many registers
- Very inefficient instruction coding
- Poor support for subroutine calls and returns
- Instruction set becomes as complicated as CISC CPU's

The eForth Approach

The eForth Approach

- Minimal instruction set:
  - Designs scalable from 16 to 64 bits
- Dual stack architecture:
  - Return stack for nested return addresses
  - Data stack for nested parameter lists
- Compute before execution:
  - All instructions executes in 1 clock cycle
- Minimized subroutine call and returns:
  - Support modular and structured programs
  - Seamless integration of high level programming language

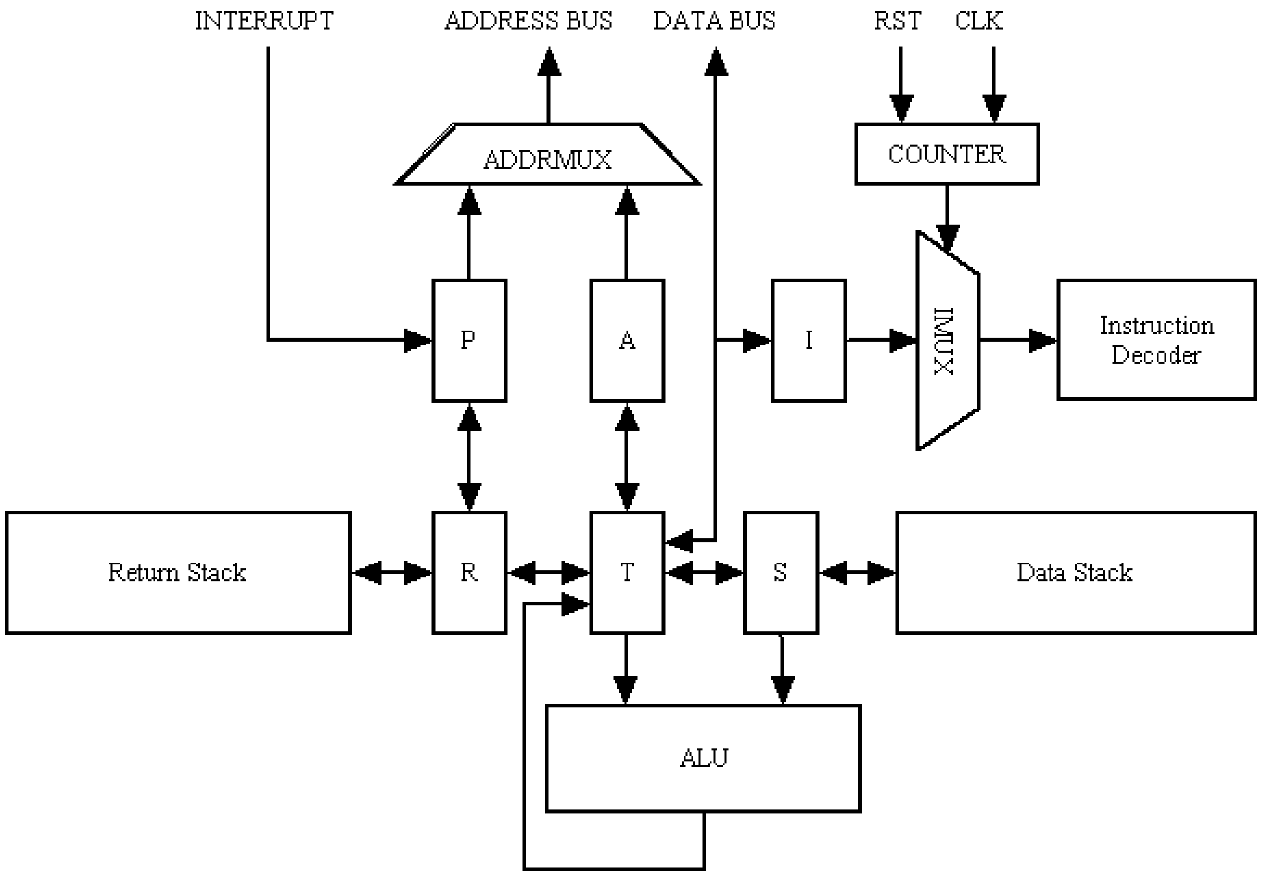
eP Series CPU Block Diagrams



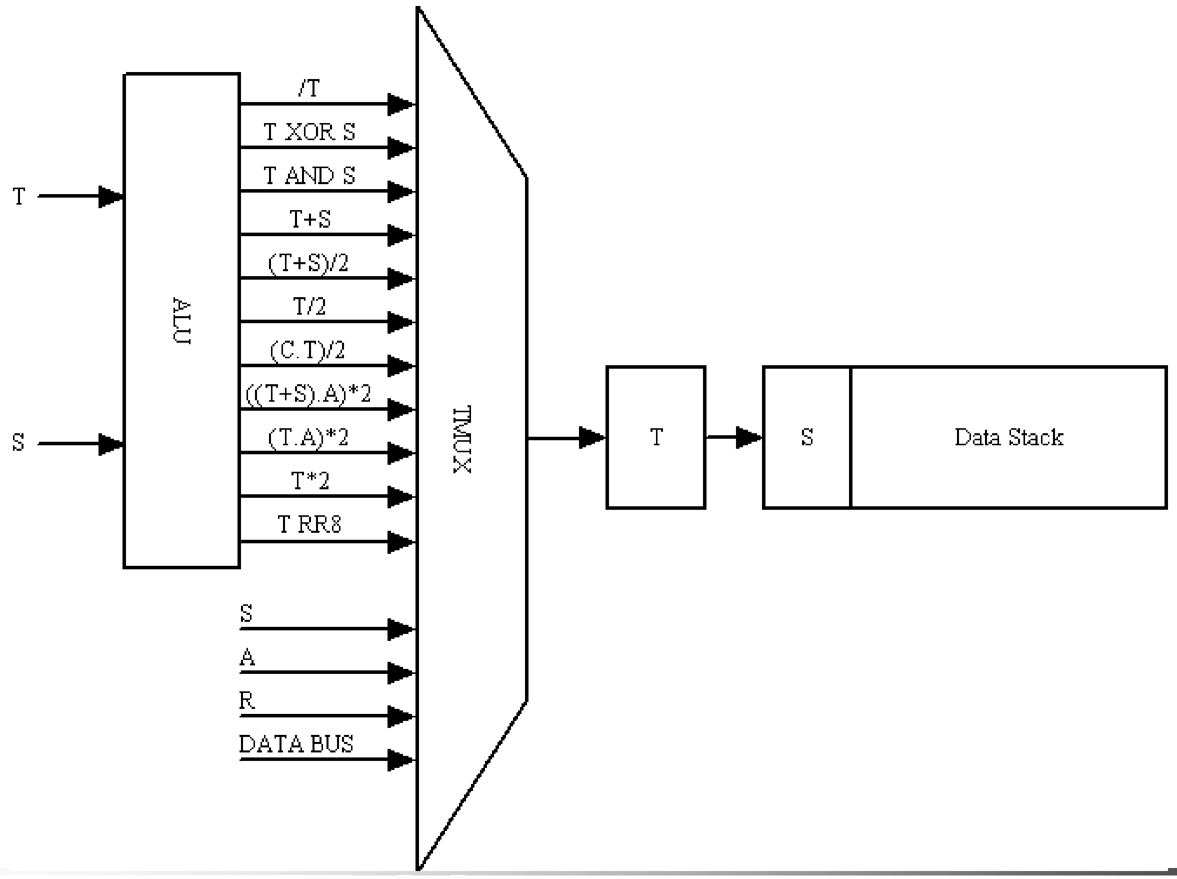
eP Series CPU Block Diagrams

- CPU architectural View
- ALU and data processing chain
- Program and data memory address multiplexer
- Return address processing chain
- Instruction execution finite state machine

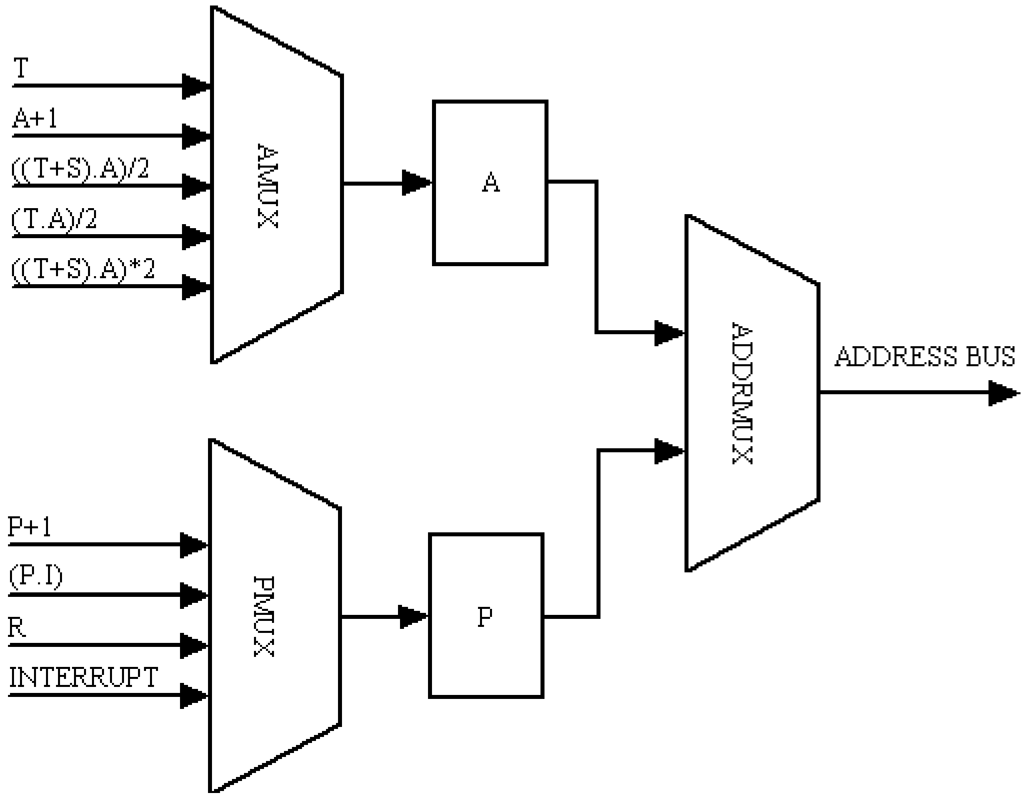
CPU Architectural View



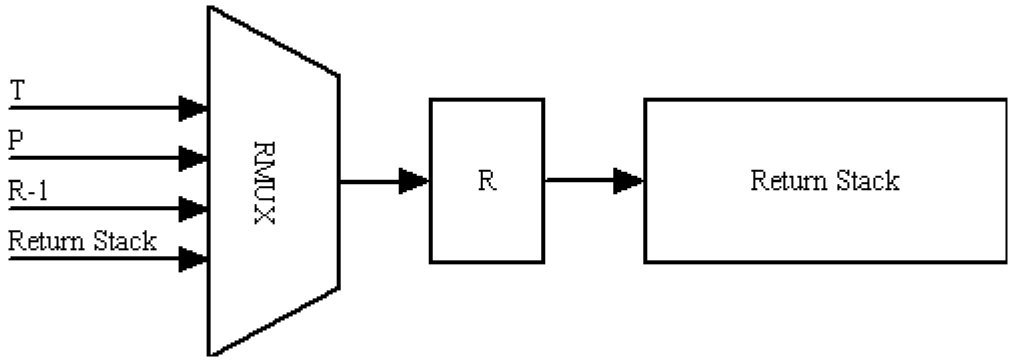
ALU and Data Processing Chain



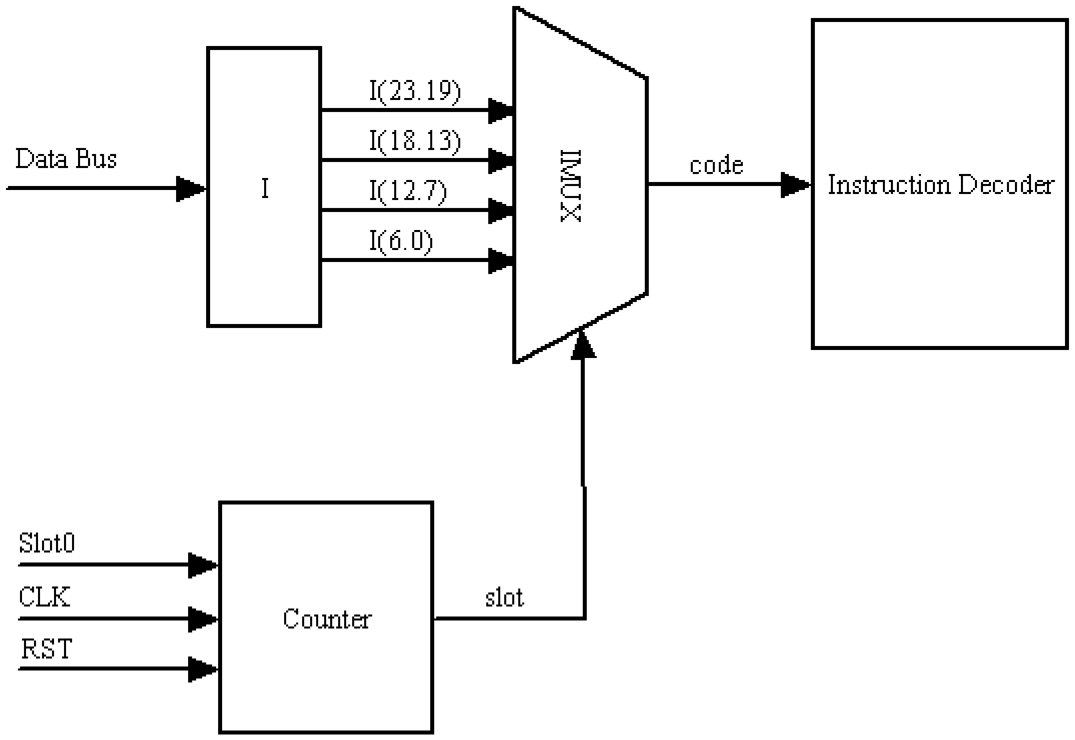
Program and Data Memory Mux



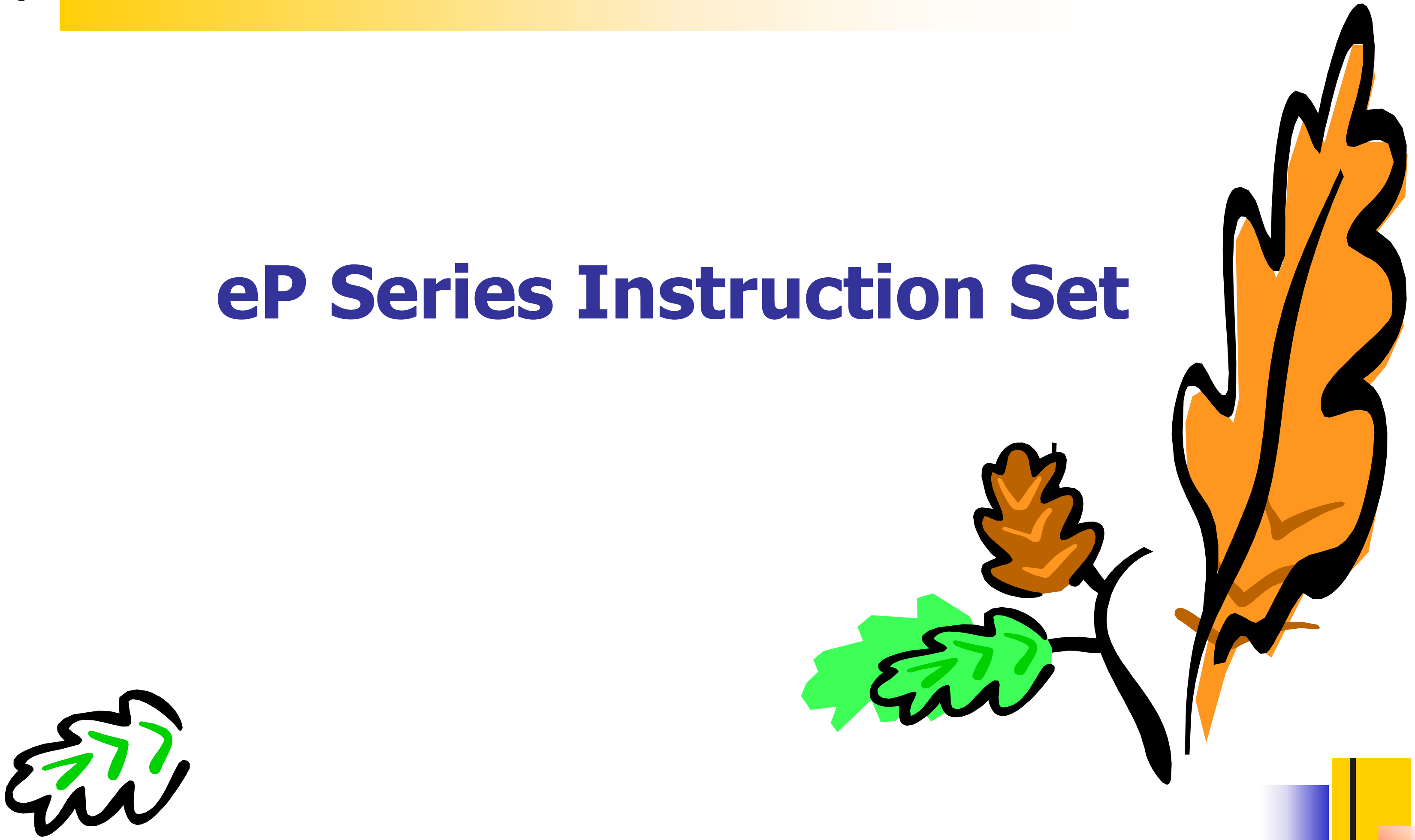
Return Address Processing Chain



Instruction Execution FSM



eP Series Instruction Set



ePSeries Instruction Set

- Only 28 orthogonal instructions
- Encoded in 5 bit fields
- Easily expandable for specific applications
- 4 Types of instructions:
  - 6 Program transfer instructions
  - 5 Memory access instruction
  - 9 ALU instructions
  - 8 Register and stack instructions

Program Transfer Instructions

BRA	Branch always
RET	Return from subroutine
BZ	Branch on zero
BC	Branch on carry
CALL	Call subroutine
NEXT	Loop until R is 0

Memory Access Instructions

LD	Load from memory
LDP	Load from memory and increment A register
LDI	Load immediate value
ST	Store to memory
STP	Store to memory and increment A register

ALU Instructions

ADD	Add S to T
AND	AND S to T
XOR	XOR S to T
COM	Complement T
SHR	T shift to right
SHL	T shift to left
RR8	T rotate right by 8 bits
MUL	Multiplication step
DIV	Division step

Register and Stack Instructions

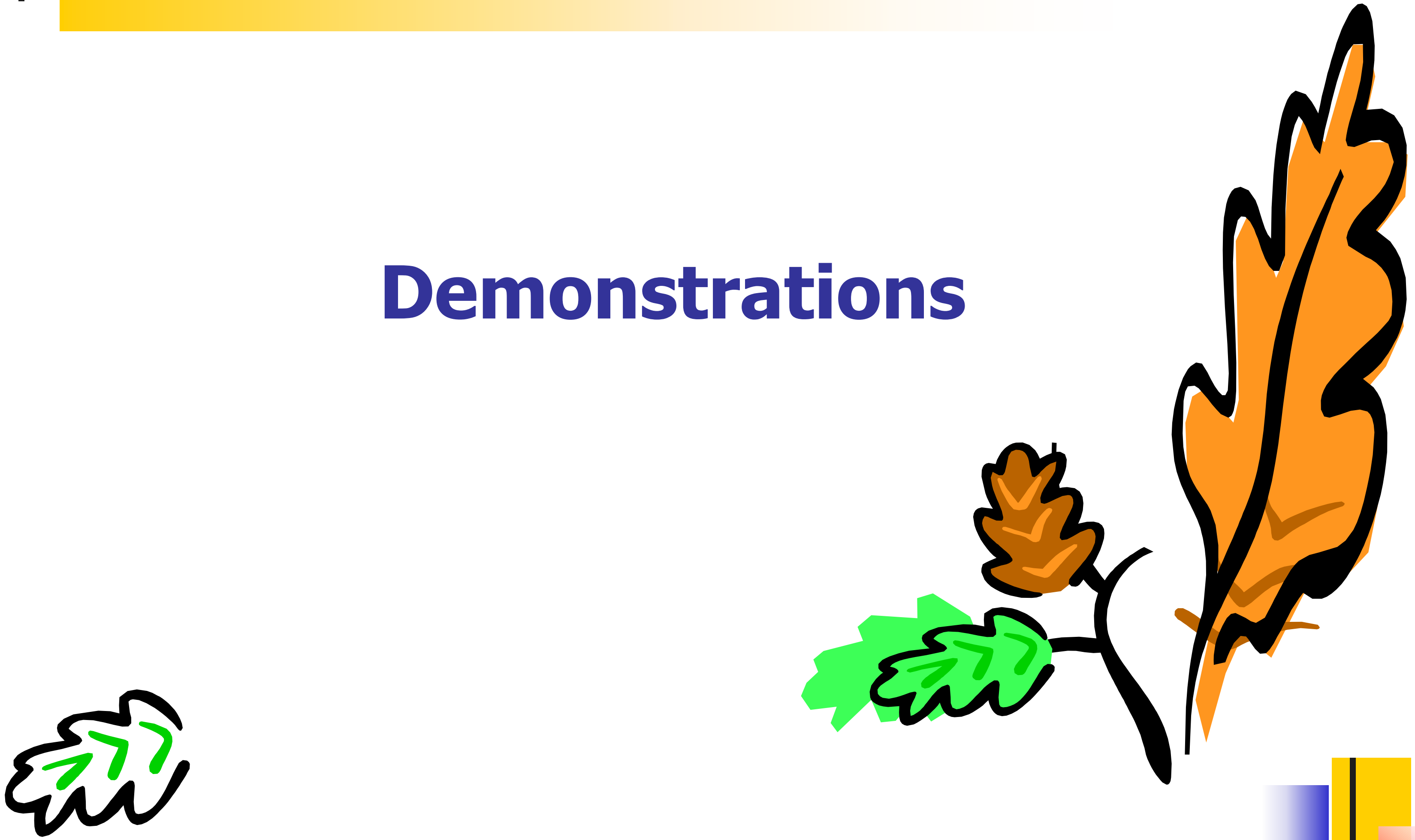
PUSHS	Duplicate T to S
POPS	Pop S to T
PUSHR	Push T to R
POPR	Pop R to T
OVER	Duplicate S over T
LDA	Load A to T
STA	Store T to A
NOP	

eForth Operating System

eForth Operating System

- Ideal for SOC and embedded systems
- Serial Port for human interface
- Command interpreter
- High level command compiler
- Low level assembler
- Debugging utilities

Demonstrations



Demonstrations

- VLSI implementation of eP8 chip
- FPGA implementation of eP24
- FPGA implementation of eP32
- Metacompiler of eForth operating system for eP chips
- Software simulator of eP chips

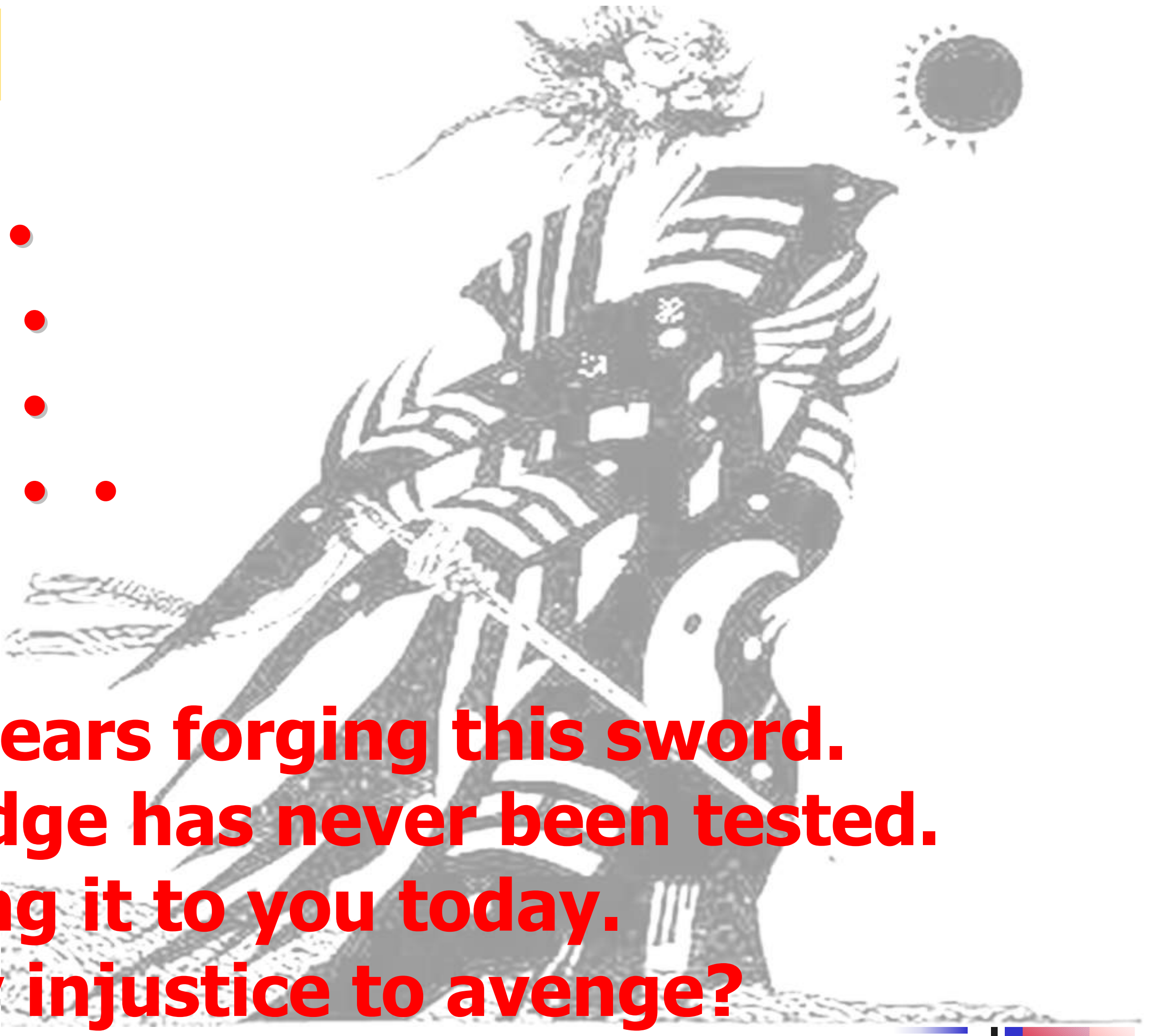
Questions and Answers

Concluding Remarks

Concluding Remarks

- Minimized instruction set allows CPU scalable from 16 to 64 bits
- Optimized CPU design allows fast execution of instructions
- Optimized subroutine call/return allows modular and structured programming
- Integrated operating system allows efficient SOC system design and integration

● ● ● ●



● ● ● ● ● ● ●
● ● ● ● ● ●
● ● ● ● ● ●
● ● ● ● ● ● ●

I spent 10 years forging this sword.
It's sharp edge has never been tested.
I am showing it to you today.
Is there any injustice to avenge?

Thank you very much!