V810 Seminar

February 21, 1995

NEC

NEC Electronics Inc.
NEC Corporation
V810 Introduction

V810 Architecture

Programming Tips & Optimization
CISC vs. RISC

CISC
= Complex Instruction
  Set Computer

RISC
= Reduced Instruction
  Set Computer

V810 Seminar
V-1123-0295-WW03

NEC
Load/Store Architecture

CISC

add _mem1, _mem2

RISC (V810 etc.)

load _mem1_disp[rBase], rX
load _mem2_disp[rBase], rY
add rX, rY
store rY, _mem2_disp[rBase]
Code Size Efficiency

![Bar Chart]

V810 Seminar
V-1123-0295-WW05

NEC
Pipeline

Non-pipeline

Instruction: IF | ID | EX | MA | WB

Pipeline

Instruction1: IF | ID | EX | MA | WB

Instruction2: IF | ID | EX | MA | WB

Instruction3: IF | ID | EX | MA | WB

Instruction4: IF | ID | EX | MA | WB

Instruction5: IF | ID | EX | MA | WB

parallel operation of 5-instruction

IF: Instruction
ID: Instruction
EX: Execution
MA: Memory
WB: Write Back

V810 Seminar
V-1123-0295-WW06
Cache Memory

High-performance; 1 clock access

Parallel Operation; instruction/data flow

CPU
- Instruction Cache
- Execution Unit

Memory
- SRAM
- DRAM
- PROM

more than 2 clock access
1 clock access

CPU
- Instruction Cache
- Execution Unit

Memory
- SRAM
- DRAM
- PROM

inst. fetch
data access

competition
parallel operation

V810 Seminar
V-1123-0295-WW07

NEC
Fast Interrupt Response

V810

Interrupt Response

PC
PSW
EIPC
EIPSW

FFFF FEn0

(cache hit)

mov rn, rm

reti

Vector Table

14Clocks (when INT handler is in cache)
V810 Architecture
Outline of Architecture

- 1K-byte instruction cache memory
- 1 clock pitch pipeline
- 16-bit/32-bit instruction length
- 32 general-purpose registers (32-bit)
- 4G-byte linear address space
- Register/flag hazard interlocked by hardware
- Floating-point operation instructions (IEEE-754)
- Bit string instructions
- 16-levels of high-speed interrupt responses
Register Set

Program Registers

- r0: Zero Register
- r1: Reserved for Address Generation
- r2: Handler Stack Pointer (hp)
- r3: Stack Pointer (sp)
- r4: Global Pointer (gp)
- r5: Text Pointer (tp)
- r6 - r25: Unused
- r26: String Destination Bit Offset
- r27: String Source Bit Offset
- r28: String Length
- r29: String Destination
- r30: String Source
- r31: Link Pointer (lp)

PC

System Registers

- 31: Exception/Interrupt PC (EIPC)
- 30: Exception/Interrupt PSW (EIPSW)
- 29: Fatal Error PC (FEPC)
- 28: Fatal Error PSW (FEPSW)
- 27: Exception Cause Register (ECR)
- 26: Program Status Word (PSW)
- 25: Processor ID Register (PIR)
- 24: Task Control Word (TKCW)
- 23: Cache Control Word (CHCW)
- 22: Address Trap Register (ADTRE)
Data Type

Integer/Unsigned Integer

Byte (B)

Halfword (H)

Word (W)

Floating Point number

Bit String

V810 Seminar
V-1123-0295-WW12

NEC
Data Alignment

- Little Endian
- Data must align to their length

<table>
<thead>
<tr>
<th>Offset</th>
<th>MSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000h</td>
<td></td>
<td>word</td>
</tr>
<tr>
<td>0004h</td>
<td>half-word</td>
<td>half-word</td>
</tr>
<tr>
<td>0008h</td>
<td>byte</td>
<td>byte</td>
</tr>
</tbody>
</table>
Instruction Alignment

- Little Endian
- Half-word (16-bit) alignment

V810 Seminar
V-1123-0295-WW14

NEC
# Instruction Set

<table>
<thead>
<tr>
<th>Category</th>
<th>Function</th>
<th>Category</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data transfer</td>
<td>General reg $\leftrightarrow$ General reg</td>
<td>Bitstring</td>
<td>Move, And, Not, Or, Exclusive-Or, Search</td>
</tr>
<tr>
<td></td>
<td>General reg $\leftrightarrow$ Memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I/O</td>
<td>Input, Output</td>
<td>Floating</td>
<td>Add, Sub, Mul, Div, Compare, Convert</td>
</tr>
<tr>
<td>Arithmetic / Logical</td>
<td>Signed/ Unsigned add, sub, mul, div</td>
<td>Branch</td>
<td>Jump, Conditional branch, Jump and link</td>
</tr>
<tr>
<td></td>
<td>Compare</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>And, Or, Not, Exclusive-Or</td>
<td>Others</td>
<td>Trap, Return from interrupt, Nop, Halt</td>
</tr>
<tr>
<td>Shift</td>
<td>Logical shift, Arithmetic shift</td>
<td></td>
<td>Compare and exchange</td>
</tr>
<tr>
<td>System</td>
<td>System Register load / store</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Execution Clock

<table>
<thead>
<tr>
<th>Move</th>
<th>Movea</th>
<th>Branch</th>
<th>Bitstring</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov</td>
<td>mova</td>
<td>jmp</td>
<td>search</td>
</tr>
<tr>
<td>load/store</td>
<td></td>
<td>jr</td>
<td></td>
</tr>
<tr>
<td>ld</td>
<td></td>
<td>jal</td>
<td></td>
</tr>
<tr>
<td>st</td>
<td></td>
<td>bcc</td>
<td></td>
</tr>
<tr>
<td>Integer/logical operation</td>
<td>op reg, reg</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>op imm, reg</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mul</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td></td>
<td>div</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td>Shift</td>
<td>sha</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>shl, shr</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Floating operation</td>
<td>addf.s</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td></td>
<td>subf.s</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mulf.s</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td></td>
<td>divf.s</td>
<td>44</td>
<td></td>
</tr>
</tbody>
</table>

This value shows the case that the same instructions are executed

* No hazard and cache hit
** Clock for word data
*** 16-bit external data bus
Instruction Format

Format -1

<table>
<thead>
<tr>
<th>15</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>opcode</td>
<td>reg2</td>
</tr>
</tbody>
</table>

:reg-reg ops.

Format -2

<table>
<thead>
<tr>
<th>15</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>opcode</td>
<td>reg2</td>
</tr>
</tbody>
</table>

:imm-reg ops.

Format -3

<table>
<thead>
<tr>
<th>15</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>opcode</td>
<td>cond</td>
</tr>
</tbody>
</table>

:Conditional branch

Format -4

<table>
<thead>
<tr>
<th>15</th>
<th>31</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>opcode</td>
<td>disp26</td>
<td></td>
</tr>
</tbody>
</table>

:Medium-range jump

Format -5

<table>
<thead>
<tr>
<th>15</th>
<th>31</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>opcode</td>
<td>reg2</td>
<td>reg1</td>
</tr>
</tbody>
</table>

:Three operand ops.

Format -6

<table>
<thead>
<tr>
<th>15</th>
<th>31</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>opcode</td>
<td>reg2</td>
<td>reg1</td>
</tr>
</tbody>
</table>

:Load/Store ops.

Format -7

<table>
<thead>
<tr>
<th>16</th>
<th>31</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>opcode</td>
<td>reg2</td>
<td>reg1</td>
</tr>
</tbody>
</table>

:Bitstring/Floating ops.

V810 Seminar
V-1123-0295-WW17

NEC
Offset Addressing Mode

- Load / Store

\[ \text{ld } 16\text{bit}\{\text{base}\},\text{reg} \]
32-bit Immediate Load

- Immediate load

- movhi imm, reg
- movea imm, reg

Diagram showing bit distribution with 31, 16, 15, and 0 positions.
Function Call Range

- Function call

```
jal _func
```

Diagram:
- PC
- +32MB
- -32MB
# Flag Operation

<table>
<thead>
<tr>
<th>Operation</th>
<th>CY</th>
<th>OV</th>
<th>S</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov, movea movhi, ld, st, in, out</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>add, addi, sub, cmp</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>mul, div, mulu, divu</td>
<td>—</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>and, or, xor, not, andi, ori, xori</td>
<td>—</td>
<td>0</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>shl, shr, sar</td>
<td>*</td>
<td>0</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>jmp, jr, jal, Bcond</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

—: Not affected    *: Affected    0: Cleared to 0
Load / In

- **Load**
  --> sign extension

- **In**
  --> zero extension
Function Call

jal op.: return address is saved into r31 register

_vfunc1:
add -imm, r3*
st.w r31, disp[r3]*

stack frame generation
push return address

_vfunc2:
add -imm, r3*
st.w r31, disp[r3]

ld.w disp[r3], r31*
add imm, r3
jmp [r31]

pop return address
stack frame release
Bitstring Operation

- **Search**
  - Search for the first 0 or 1 from the specified bit
  - downward
  - upward

- **Move**
  - Source
  - Destination
  - Move/Move after Not

- **Logical operation (BitBLT operation)**
  - Source
  - Destination
  - and/or/xor
  - Destination

---

**V810 Seminar**

V-1123-0295-WW24

NEC
Floating Point Operation

- Single precision floating point operation (IEEE-754 standard)
  - Performance = 0.9 M FLOPS (25MHz)
  - Add, Sub, Mul, Div, Compare (26-44 execution clocks)
  - Conversion (32 bit float ↔ 32 bit integer)
  - Floating data are handled in General Registers
Interrupt/Exception Flow
Interrupt:
- Maskable interrupt (16 levels)
- Non-maskable interrupt

Exception:
- Double exception
- Trap instruction
- Address trap
- Reserved op code
- Zero division
- Floating operation exception (6 types)
- Reset
Maskable Interrupt

Maskable interrupt (INT) occurs

- If PSW.NP = 1, ignored.
- If PSW.EP = 1, ignored.
- If PSW.ID = 1, ignored.
- If Interrupt Level < Interrupt enable level, ignored.
- If Interrupt Level ≥ Interrupt enable level, ignored.

- EIPC ← restore PC
- EIPSW ← PSW
- ECR.EICC ← exception code
- PSW.EP ← 1
- PSW.ID ← 1
- PSW.AE ← 0
- Sets PSW.I3-10

Jumps to handler address
Non-Maskable Interrupt

Non-maskable interrupt (NMI) occurs

- PSW.NP: 1

Interrupt request is internally held. Processing is started when NP of PSW is cleared to 0.

- FEPC: restore PC
- FEPSW: PSW
- ECR.FECC: exception code
- PSW.NP: 1
- PSW.ID: 1
- PSW.AE: 0

Jumps to handler address (Address FFFFFFD0H)
Exception Processing

Exception occurs

PSW.NP

0

PSW.EP

1
Double exception

EIPC  \( \rightarrow \) restore PC
EIPSW  \( \rightarrow \) PSW
ECR.EICC  \( \rightarrow \) exception code
PSW.EP  \( \rightarrow \) 1
PSW.ID  \( \rightarrow \) 1
PSW.AE  \( \rightarrow \) 0

Jumps to handler address

Fatal exception

Machine fault status
Address 00000000H \( \leftarrow \) source code
Address 00000004H \( \leftarrow \) current PSW
Address 00000008H \( \leftarrow \) current PC

Stop

FEPC  \( \rightarrow \) restore PC
FEPSW  \( \rightarrow \) PSW
ECR.FECC  \( \rightarrow \) exception code
PSW.NP  \( \rightarrow \) 1
PSW.ID  \( \rightarrow \) 1
PSW.AE  \( \rightarrow \) 0

Jumps to handler address
( Address FFFFFFFD0H )
Return from Exception/Interrupt

RET instruction

PSW.NP

0

1

PC ← EIPC
PSW ← EIPSW

Jumps to PC

PC ← FEPC
PSW ← FEPSW

Jumps to PC
Important H/W Issues
Interlock support (load/store/flag Interlock).
- Hazard detection & interlock by H/W.
- Transparent to assembler programming & debugging.
- Small code size by reducing excessive instruction.
- Better performance by inserting effective instruction.
Load/Store Interlock

- Changing base register just before load/store.

**General RISC**
- assembler inserts
- software wait.

**V800**
- hardware detects hazard
- and stalls pipeline.

**ex.**
- add r3,r6
- nop
- ld.w disp[r6],r10

**ex.**
- add r3,r6
- ld.w disp[r6],r10
Flag Interlock

Conditional branch just after flag modification.

**General RISC**
Assembler inserts software wait.

- **ex.**
  - `cmp r6,r10`
  - `nop`
  - `bz`

**V800**
Hardware detects hazard and stalls pipeline.

- **ex.**
  - `cmp r6,r10`
  - `bz`
Cache Implementation

Capacity: 1K bytes
Mapping method: Direct mapping
Block size: 8 bytes
Subblock size: 4 bytes

Memory address

TAG

Index

Offset

Entry 0

TAG31-TAG10

Entry 1

Tag memory (ICHT27-ICHT0)

Entry 127

Valid bit (1 bit for each 4 bytes)

NECRV (reserved by NEC for debugger, must be zero in normal operation)
Cache Tips

Locality of program execution is very important.
- Loop (with many loop count) -> very good
- Key for performance (ex. INT handler) -> good
- Rarely executing -> poor
- Executing only once (ex. boot routine) -> very poor

Can control by CHCW (cache control word) register.
- Enable/Disable
- Clear all/part
- Dump/Restore to/from memory
NEC
NEC Electronics Inc.