Outline

- Introduction
- Architecture of a RISC Processor
- Implementation

Architecture

- Software visible registers
- Memory Addressing
- The instruction set
- The exception / reset mechanism

Architecture

Software visible registers

Registers that can be manipulated (written or read) in the assembly language

Software Visible Registers

- 32 common 32-bit registers

Integer Registers

R_0 …. R_{31}

Addressed from their number

Software Visible Registers

- R_0 : The Trash Register

  A value written into R_0 is lost
  R_0 contains always 0

- R_{31} : The Link Register

  When a subroutine is called, the return address is saved into R_{31}

Software Visible Registers

- Two 32-bit registers : Hi and Lo

  Used by multiply and divide instructions

Multiply	Hi	32 most significant bits
	Lo	32 least significant bits
Divide	Hi	Remainder
	Lo	Quotient

Software Visible Registers

- Six 32-bit special registers : Coprocessor Registers

  Required to implement an operating system

Status	Status Register
Cause	Cause Register (cause of exceptions)
Ebase	Exception Base Register
Epc	Exception Program Counter (return address in case of exception)
Eepc	Error Exception Program Counter (return address in case of exception)
BadVAddr	Bad Virtual Address Register (illegal memory address)

Architecture

- Software visible registers
- Memory Addressing
- The instruction set
- The exception / reset mechanism

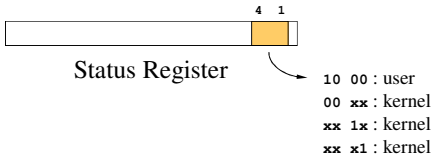
Memory Addressing

- 32-bit address ⇨ 4 Gbytes of memory space
- Read / Write operations
- 3 types of data : Byte
  Half-word (2 bytes)
  Word (4 bytes)

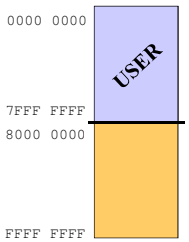
Memory Addressing

- The processor can operate under 2 modes
  User / Kernel

The current mode is defined by the Status Register



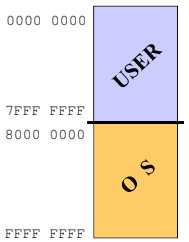
Memory Addressing



The memory space is divided into 2 parts

In *User* mode the processor can access only the addresses ranged from `0000 0000` to `7FFF FFF`

Memory Addressing

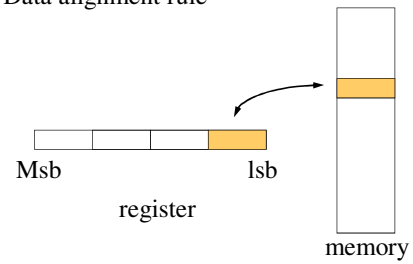


The OS protects the hardware against an error in a user program

This frontier protects the OS

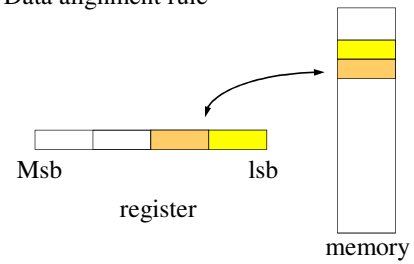
Memory Addressing

- Data alignment rule



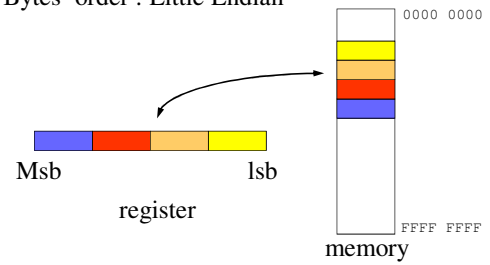
Memory Addressing

- Data alignment rule



Memory Addressing

- Bytes' order : Little Endian



Memory Addressing

- Address alignment rule

The address of an object of N bytes must be multiple of N

Address of a Word	⇨	multiple of 4
Address of a Half-Word	⇨	multiple of 2
Address of a Byte	⇨	multiple of 1