Computer Architecture

Discussion 4

Minye Wu
wumy@shanghaitech.edu.cn

Mar. 15 2016

Discussion

- What is SYSCALL ?
- How SYSCALL works?

MIPS interrupts

- Interrupts are events that demand the processor’s attention
- Must be handled without affecting any active programs.
- Since interrupts can happen at any time, there is no way for the active programs to prepare for the interrupt.

when an interrupt occurs

- **When an interrupt occurs, your processor may perform the following actions:**
- move the current PC into another register, call the EPC
- record the reason for the exception in the Cause register
- automatically disable further interrupts from occurring, by left-shifting the Status register
- change control (jump) to a hardwired interrupt handler address
- **To return from a handler, your processor may perform the following actions:**
- move the contents of the EPC register to the PC.
- re-enable interrupts, by right-shifting the Status register

Cause register

The Cause register is a 32-bit register, but only certain fields on that register will be used.
Bits 1 down to 0 will be set to describe the cause of the last interrupt/exception.

Number	Name	Description
00	INT	
01	IBUS	Instruction bus error (invalid instruction)
10	OVF	Arithmetic overflow
11	SYSCALL	System call

Status register

- The status register is also a 32-bit register.
- Bits 3 down to 0 will define masks for the three types of interrupts/exceptions.
- If an interrupt/exception occurs when its mask bit is current set to 0, then the interrupt/exception will be ignored.

Bit	Interrupt/exception
3	INT
2	IBUS
1	OVF
0	SYSCALL

Back to SYSCALL functions in MARS

- **How to use SYSCALL system services**

- Step 1. Load the service number in register $v0.
- Step 2. Load argument values, if any, in $a0, $a1, $a2, or $f12 as specified.
- Step 3. Issue the SYSCALL instruction.
- Step 4. Retrieve return values, if any, from result registers as specified.

Service	Code in $v0	Arguments	Result
print integer	1	$a0 = integer to print	
print float	2	$f12 = float to print	
print double	3	$f12 = double to print	
print string	4	$a0 = address of null-terminated string to print	
read integer	5		$v0 contains integer read
read float	6		$f0 contains float read
read double	7		$f0 contains double read
read string	8	$a0 = address of input buffer $a1 = maximum number of characters to read	……

http://courses.missouristate.edu/KenVollmar/mars/Help/SyscallHelp.html

Read and print an integer in MARS

- li $v0, 5
- syscall
- add $t0,$v0,$zero
- li $v0, 1
- add $a0, $t0, $zero
- syscall

Exercise

- Input a string and output the substring that begins with the second character.

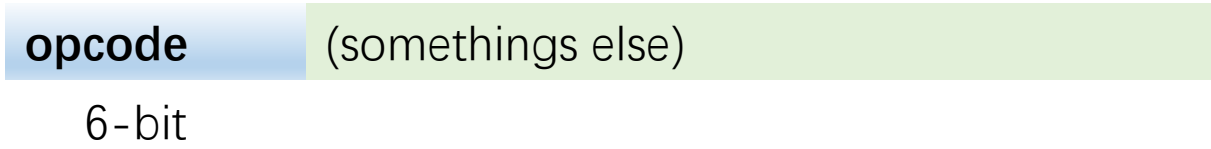
```
.data
STRING: .word 0:10
.text
li $v0,8
la $a0,STRING
li $a1,30
syscall
li $v0,4
la $a0,STRING
add $a0,$a0,1
syscall
```

About project

- **Step 0: Obtaining the Files**
- **Step 1: Building Blocks**
- **Step 2: SymbolTable**
- **Step 3: Instruction Translation**
- **Step 4: Pseudoinstruction Expansion**
- **……**

MIPS Instruction Formats

- R\I\J\FR\FI Instructions



https://en.wikibooks.org/wiki/MIPS_Assembly/Instruction_Formats
http://www.mrc.uidaho.edu/mrc/people/jff/digital/MIPSir.html

Why we need two passes ?

What is the address?

```
        .data
fibs:   .word   0 : 12          # "array" of 12 words to contain fib values
size:   .word   12              # size of "array"
        .text
        la   $t0, fibs          # load address of array
        la   $t5, size          # load address of size variable
        lw   $t5, 0($t5)        # load array size
        li   $t2, 1             # 1 is first and second Fib. number
        add.d $f0, $f2, $f4
        sw   $t2, 0($t0)        # F[0] = 1
        sw   $t2, 4($t0)        # F[1] = F[0] = 1
        addi $t1, $t5, -2       # Counter for loop, will execute (size-2) times
loop:   lw   $t3, 0($t0)        # Get value from array F[n]
        lw   $t4, 4($t0)        # Get value from array F[n+1]
        add  $t2, $t3, $t4      # $t2 = F[n] + F[n+1]
        sw   $t2, 8($t0)        # Store F[n+2] = F[n] + F[n+1] in array
        addi $t0, $t0, 4        # increment address of Fib. number source
        addi $t1, $t1, -1       # decrement loop counter
        bgtz $t1, loop          # repeat if not finished yet.
        la   $a0, fibs          # first argument for print (array)
        add  $a1, $zero, $t5    # second argument for print (size)
        jal  print              # call print routine.
        li   $v0, 10            # system call for exit
        syscall                 # we are out of here.
```