lecture 11

MIPS assembly language 4

- functions
- MIPS stack

February 15, 2016

MIPS registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Register Number</th>
<th>Usage</th>
<th>Preserved on call</th>
</tr>
</thead>
<tbody>
<tr>
<td>zero</td>
<td>0</td>
<td>the constant value 0</td>
<td>n.a.</td>
</tr>
<tr>
<td>s1</td>
<td>1</td>
<td>reserved for the assembler</td>
<td>n.a.</td>
</tr>
<tr>
<td>s0-s7</td>
<td>2-7</td>
<td>value for results and expressions</td>
<td>no</td>
</tr>
<tr>
<td>s8-s15</td>
<td>8-15</td>
<td>arguments (procedures/functions)</td>
<td>yes</td>
</tr>
<tr>
<td>a0-a7</td>
<td>16-23</td>
<td>temporaries</td>
<td>no</td>
</tr>
<tr>
<td>a8-a9</td>
<td>24-25</td>
<td>saved</td>
<td>yes</td>
</tr>
<tr>
<td>a10-a17</td>
<td>26-33</td>
<td>more temporaries</td>
<td>no</td>
</tr>
<tr>
<td>gp</td>
<td>28</td>
<td>reserved for the operating system</td>
<td>n.a.</td>
</tr>
<tr>
<td>sp</td>
<td>29</td>
<td>stack pointer</td>
<td>yes</td>
</tr>
<tr>
<td>fp</td>
<td>30</td>
<td>frame pointer</td>
<td>yes</td>
</tr>
<tr>
<td>lr</td>
<td>31</td>
<td>return address</td>
<td>yes</td>
</tr>
</tbody>
</table>

parent must:
- save in Memory any registers it will need later (after return)
- provide arguments to child
- provide return address to child (so child can jump back when done)
- branch to child

child must:
- allocate space for local variables (registers, memory) and not write over parent's data (registers, memory)
- compute value and return it to parent
- branch back to parent

Jump and Link, Jump register

main:
jal myfunction
jal

myfunction:
jr $ra

functions in C

int myfunction(int j) {
    int m, n;
    m = myfunction(n);
    return k;
}

Call

parent must:
- branch to child

child must:
- branch back to parent

Provide argument(s), return value(s)

argument registers
$2, $3, $4, $5

return value registers
$0, $1
Encapsulation Problem

- Author of parent function might not know which $s$ and $t$ registers the child uses.
- Author of child function might not know which $s$ and $t$ registers the parent uses.

MIPS register conventions (polices)

- **Parent (caller)**
  - Assume that $s0, ..., s7$ will contain some values before and after call.
  - Don't assume that $t0, ..., t7$ will contain some values after call.
  (If parent will need values in $t0, ..., t7$ after call, then these values must be stored in memory prior to call, and loaded after call.)

- **Child (callee)**
  - Assume that $s0, ..., s7$ are not being used by parent.
  - Assume that $t0, ..., t7$ are being used by parent.

How and where do functions store register values in memory?

<table>
<thead>
<tr>
<th>$s0$</th>
<th>$s1$</th>
</tr>
</thead>
</table>

Memory

Stack frames

User data

Heap

User instructions

Function arguments, local variables, return address, etc.
main stores temporary registers before calling myfunction. (Here we assume variables m and n use $t0 and $t1.)

myfunction stores previous values of any save registers that it will use. (Here we assume variable k uses $s0.)

Stack pointer

Stack pointer (register $sp = $29) contains the lowest address of the stack.

When a function is both a child and a parent, it must store the return address (to its own parent) before it calls its child. It also needs to store $a registers (from parent) that it modifies. (See conventions on slide 2.)
Example: recursion

```c
int sumton (int n) { // n > 0
    if (n == 0)
        return 0;
    else
        return n + sumton(n-1);
}
```

**Frame Pointer ($sp$)**

- Sometimes the size of the stack frame is not fixed.
- It may be useful to keep track of where frame starts.

**Announcements**

A2 is due next Sunday at midnight.

A3 will be posted soon and will be due on the last day of Reading Week (~3 weeks from now)

A4 (last one) will be posted in mid-March and due at end of March.

Last lecture is April 13. Final exam is April 26 (tent.)