Some Ideas for Stack Computer Design

Phil Koopman

United Technologies Research Center

OVERVIEW

- How to pipeline a stack computer
  - RISC = Pipelined
  - Resurrection of the skip instruction
- Making the most of on-chip memory
  - Memory hierarchies for real-time control
- Stack-based execution for C
  - Program size can be smaller

HOW TO PIPELINE A STACK COMPUTER -- 1

- Traditional pipeline
  - Assumes access into register file takes time
  - Assumes writeback of result into register file takes time
  - Keeping branch delay to one slot is challenging
  - Has a load delay slot

HOW TO PIPELINE A STACK COMPUTER -- 2

- Proposed stack pipeline
  - Access to TOS is free for read and writes
  - Latency of operand access stage avoided
  - Can support auto-increment/decrement accesses
  - Can exploit two paths to memory
  - Has a load delay slot

SKIP INSTRUCTIONS

- Branch delays are a problem with pipelines
  - Delay slot on subroutine call questionable for Forth
  - Filling branch delay slot difficult on stack machines
- Using skip instructions can solve the problem

- Only branch type is CALL (1 bit controls address generation)
- Conditional branch is a skip over a call instruction -- buries branch delay slot into the skipped instruction

<table>
<thead>
<tr>
<th>I-FETCH</th>
<th>SKIP</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL</td>
<td>I-FETCH</td>
<td>ALU</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BRANCH TARGET</th>
<th>OR</th>
<th>IN-LINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-FETCH</td>
<td>ALU</td>
<td>MEMORY R/W</td>
</tr>
</tbody>
</table>

MAKING THE MOST OF ON-CHIP MEMORY (for real-time control)

- Caches inappropriate for many real-time systems
- On-chip memory much faster than off-chip accesses
- Use large on-chip memory for user-controlled storage
  - Microcode ROM/RAM
  - Instruction ROM/RAM
  - Small data memory
- Idea: on-chip program memory to provide split-bus access without extra pins

STACK-BASED EXECUTION FOR C

- IF:
  - Memory space is limited
  - Program size matters more than ultimate speed
- THEN:
  - Stack-relative addressing increases opportunities for code-sharing
  - Assignment of registers makes two identical code sequences compile to different binary images
  - A compiler can compress code using stack-based execution and shared code sequences
- A C compiler can produce passable Forth-style code
  - Depends on locality of reference to C variables & coding style