Emulation: Binary Translation

CSCE496/896: Embedded Systems Design and Implementation
Witawas Srisa-an
Interpretation

Mapping a source binary instruction to an equivalent HLL code

- e.g. binary instructions to C functions
  - 45 alpha instructions for 1 IA-32 instruction
  - 30 alpha instructions for 1 micro-OP instruction
Binary Translation

Mapping each source binary instruction to its own customized target code

- binary to binary
- lower ratio between source and target instructions
  - 4.4 Alpha instructions per 1 IA-32 instruction
  - 2.1 Alpha instructions per 1 micro-OP instruction
- more efficient with opportunities for optimizations
Binary Translation

Direct Threaded Interpretation

Binary Translator
Binary Translation

IA32:

```assembly
addl 4(%eax),%edx
movl %edx,4(%eax)
add %eax, 4
```
Binary Translation

IA-32 to MIPS

Assume
r1 points to IA32 register file
r2 points to IA32 memory image
r3 contains IA32 PC value

IA32:
addl 4(%eax),%edx
movl %edx,4(%eax)
add %eax,4

MIPS:
; addl 4(%eax),%edx
lw r4,0(r1)
addi r5,r4,4
add r5,r2,r5
lw r5,0(r5)
lw r4,12(r1)
add r5,r5,r4
sw r5,12(r1)
addi r3,r3,3

MIPS:
; movl %edx,4(%eax)
lw r4,0(r1)
addi r5,r4,4
lw r4,12(r1)
add r5,r2,r5
sw r4,0(r5)
addi r3,r3,3

; addl 4(%eax),%edx
lw r4,0(r1)
addi r5,r4,4
add r5,r2,r5
lw r5,0(r5)
lw r4,12(r1)
add r5,r5,r4
sw r5,12(r1)
addi r3,r3,3

3 IA-32 instructions to 18 MIPS instructions
Binary Translation

State Mapping

Source ISA
- Register File
- Memory Image
- PC
- R1
- R2
- Rn

Target ISA
- R1
- R2
- R3
- R4
- R5
- R(n + 3)
**Binary Translation**

**IA-32 to MIPS**

Assume

- r1 points to IA32 register file
- r2 points to IA32 memory image
- r3 contains IA32 PC value
- r4 holds register %eax
- r7 holds register %edx

**IA32:**

```
addl 4(%eax),%edx
movl %edx,4(%eax)
add %eax, 4
```

**MIPS:**

```
addi r16,r4,4
add r17,r2,r16
lw r17,0(r17)
add r7,r17,r7
addi r16,r4,4
add r17,r2,r16
sw r7,0(r17)
addi r4,r4,4
addi r3,r3,9
```
**Binary Translation**

**IA-32 to MIPS**

<table>
<thead>
<tr>
<th>Assume</th>
<th>IA32:</th>
<th>MIPS:</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1 points to IA32 register file</td>
<td>addl 4(%eax),%edx</td>
<td>addi r16,r4,4</td>
</tr>
<tr>
<td>r2 points to IA32 memory image</td>
<td>movl %edx,4(%eax)</td>
<td>add r17,r2,r16</td>
</tr>
<tr>
<td>r3 contains IA32 PC value</td>
<td>add %eax, 4</td>
<td>lw r18,0(r17)</td>
</tr>
<tr>
<td>r4 holds register %eax</td>
<td></td>
<td>add r7,r18,r7</td>
</tr>
<tr>
<td>r7 holds register %edx</td>
<td></td>
<td>sw r7,0(r17)</td>
</tr>
</tbody>
</table>

3 IA-32 instructions to 7 MIPS instructions
Binary Translation

- Static translation – translate the entire executable source to executable target
- Dynamic translation – translate only a short sequence at a time
  - translated code is cached
  - frequently used code can be dynamically recompiled
Code-Discovery problem

- static precoding of the entire program is nearly impossible
- e.g. jr R1 what is the content of R1?
- where is the starting point of an instruction in CISC?
Binary Translation

Code-Location problem

inconsistency between the SPC address and TPC address

\[
\text{movl} \ %\text{eax}, 4(\%\text{esp}) \\
\text{jmp} \ %\text{eax}
\]

\[
\text{add} \ r16, r2, r11 \\
\text{lw} \ r4, 4(r16) \\
\text{jr} \ r4
\]