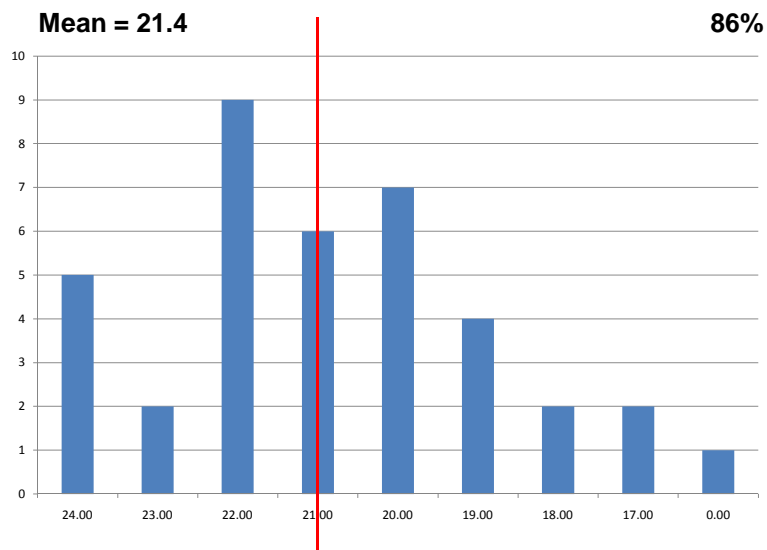


Virtual Machines

Emulation

Midterm



Emulation

- Problem: Emulate guest ISA on host ISA

Emulation

- Problem: Emulate guest ISA on host ISA
- Solution: Basic Interpretation

```
inst = code (PC)
opcode = extract_opcode (inst)
switch (opcode) {
    case opcode1 : call emulate_opcode1 ()
    case opcode2 : call emulate_opcode2 ()
    ...
}
```

Emulation

- Problem: Emulate guest ISA on host ISA
- Solution: Basic Interpretation

```

new      inst = code (PC)
         opcode = extract_opcode (inst)
         routineCase = dispatch (opcode)
         jump routineCase
...
routineCase      call routine_address
                 jump new

```

Threaded Interpretation

```

[ body of emulate_opcode1 ]
inst = code (PC)
opcode = extract_opcode (inst)
routine_address = dispatch (opcode)
jump routine_address

```

```

[ body of emulate_opcode1 ]
inst = code (PC)
opcode = extract_opcode (inst)
routine_address = dispatch (opcode)
jump routine_address

```

A Note on Extracting Opcode

- `extract_opcode (inst)`
 - Opcode may have options
 - Instruction must extract and combine several bit ranges in the machine word
 - Operands must also be extracted from other bit ranges
- Pre-decoding
 - Pre-extract the opcodes and operands for all instructions in program.
 - Put them on byte boundaries (intermediate code)
 - Must maintain two program counters. Why?

Example

`lwz r1, 8(r2)`

`add r3, r3, r1`

`stw r3, 0(r4)`

07		
1	2	08

08		
3	1	03

37		
3	4	00

Direct Threaded Interpretation

- Replace opcode with address of emulating routine

Routine_address07		
1	2	08

Routine_address08		
3	1	03

Routine_address37		
3	4	00