Principles of Computers
17th Lecture

http://d3s.mff.cuni.cz/~jezek

Pavel Ježek, Ph.D.
pavel.jezek@d3s.mff.cuni.cz
Stack Machine

- Load/store architecture
- Loads have implicit target (register on top)
- Stores have implicit source (register on top)
- Arithmetic instructions have all operands implicit (n-ary operation takes n registers from top)

CPU registers

```
? R15
? R14
? R13
? R12
? R11
? R10
? R9
? R8
? R7
? R6
? R5
? R4
? R3
? R2
? R1
? R0
```

Instruction set:

- LOAD x
- STORE x
- ADD
- SUB
- MUL
- AND
- OR
- NOT
- SHR
- SHL
- ...

PUSH (store on memory [call] stack)
POP (load from memory [call] stack)
Stack Machine

- Load/store architecture
- Loads have implicit target (register on top)
- Stores have implicit source (register on top)
- Arithmetic instructions have all operands implicit (n-nary operation takes n registers from top)

### CPU registers

| ?  | R15 |
| ?  | R14 |
| ?  | R13 |
| ?  | R12 |
| ?  | R11 |
| ?  | R10 |
| ?  | R9  |
| ?  | R8  |
| ?  | R7  |
| ?  | R6  |
| ?  | R5  |
| ?  | R4  |
| ?  | R3  |
| ?  | R2  |
| ?  | R1  |
| ?  | R0  |

### Disassembled machine code

- LOAD 2
- LOAD 4
- ADD
- LOAD 1
- LOAD 1
- ADD
- LOAD $F7
- NOT
- MUL
- ADD
- STORE x

Disassembled machine code

memory
Stack Machine

- Load/store architecture
- Loads have implicit target (register on top)
- Stores have implicit source (register on top)
- Arithmetic instructions have all operands implicit (n-ary operation takes n registers from top)

CPU registers

```
?       R15
?       R14
?       R13
?       R12
?       R11
?       R10
?       R9
?       R8
?       R7
?       R6
?       R5
?       R4
?       R3
?       R2
?       R1
?       R0
```

Disassembled machine code:

- LOAD 2
- LOAD 4
- ADD
- LOAD 1
- LOAD 1
- ADD
- LOAD $F7
- NOT
- MUL
- ADD
- STORE x
Stack Machine

- Load/store architecture
- Loads have implicit target (register on top)
- Stores have implicit source (register on top)
- Arithmetic instructions have all operands implicit (n-ary operation takes n registers from top)

CPU registers

<table>
<thead>
<tr>
<th></th>
<th>R15</th>
<th></th>
<th>R14</th>
<th></th>
<th>R13</th>
<th></th>
<th>R12</th>
<th></th>
<th>R11</th>
<th></th>
<th>R10</th>
<th></th>
<th>R9</th>
<th></th>
<th>R8</th>
<th></th>
<th>R7</th>
<th></th>
<th>R6</th>
<th></th>
<th>R5</th>
<th></th>
<th>R4</th>
<th></th>
<th>R3</th>
<th></th>
<th>R2</th>
<th></th>
<th>R1</th>
<th></th>
<th>R0</th>
</tr>
</thead>
</table>

RegStackTop

Flags

(SP)

IP

memory

LOAD 2
LOAD 4
ADD
LOAD 1
LOAD 1
ADD
LOAD $F7
NOT
MUL
ADD
STORE x
**Stack Machine**

- Load/store architecture
- Loads have implicit target (register on top)
- Stores have implicit source (register on top)
- Arithmetic instructions have all operands implicit (n-ary operation takes n registers from top)

**CPU registers**

- LOAD 2
- LOAD 4
- ADD
- LOAD 1
- LOAD 1
- ADD
- LOAD $F7
- NOT
- MUL
- ADD
- STORE x

**RegStackTop**
- Flags
- (SP)
- IP
Stack Machine

- Load/store architecture
- Loads have implicit target (register on top)
- Stores have implicit source (register on top)
- Arithmetic instructions have all operands implicit (n-nary operation takes n registers from top)

CPU registers

| ? | R15 |
| ? | R14 |
| ? | R13 |
| ? | R12 |
| ? | R11 |
| ? | R10 |
| ? | R9  |
| ? | R8  |
| ? | R7  |
| ? | R6  |
| ? | R5  |
| ? | R4  |
| ? | R3  |
| ? | R2  |
| 4 | R1  |
| 6 | R0  |

RegStackTop
Flags
(SP)
IP

memory

LOAD 2
LOAD 4
ADD
LOAD 1
LOAD 1
ADD
LOAD $F7
NOT
MUL
ADD
STORE x
Stack Machine

- Load/store architecture
- Loads have implicit target (register on top)
- Stores have implicit source (register on top)
- Arithmetic instructions have all operands implicit (n-ary operation takes n registers from top)

CPU registers

<table>
<thead>
<tr>
<th></th>
<th>R15</th>
<th>R14</th>
<th>R13</th>
<th>R12</th>
<th>R11</th>
<th>R10</th>
<th>R9</th>
<th>R8</th>
<th>R7</th>
<th>R6</th>
<th>R5</th>
<th>R4</th>
<th>R3</th>
<th>R2</th>
<th>R1</th>
<th>R0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegStackTop</td>
<td>Flags</td>
<td>(SP)</td>
<td>IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

memory

- LOAD 2
- LOAD 4
- ADD
- LOAD 1
- LOAD 1
- ADD
- LOAD $F7
- NOT
- MUL
- ADD
- STORE x
Stack Machine

- Load/store architecture
- Loads have implicit target (register on top)
- Stores have implicit source (register on top)
- Arithmetic instructions have all operands implicit (n-nary operation takes n registers from top)

### CPU registers

- RegStackTop
- Flags
- (SP)
- IP

### LOAD 2
### LOAD 4
### ADD
### LOAD 1
### LOAD 1
### ADD
### LOAD $F7
### NOT
### MUL
### ADD
### STORE x

Memory

```
LOAD 2
LOAD 4
ADD
LOAD 1
LOAD 1
ADD
LOAD $F7
NOT
MUL
ADD
STORE x
```
Stack Machine

- Load/store architecture
- Loads have implicit target (register on top)
- Stores have implicit source (register on top)
- Arithmetic instructions have all operands implicit (n-ary operation takes n registers from top)

CPU registers
- R15
- R14
- R13
- R12
- R11
- R10
- R9
- R8
- R7
- R6
- R5
- R4
- R3
- R2
- R1
- R0

RegStackTop
Flags
(SP)
IP

memory

• LOAD 2
• LOAD 4
• ADD
• LOAD 1
• LOAD 1
• ADD
• LOAD $F7
• NOT
• MUL
• ADD
• STORE x
Stack Machine

- Load/store architecture
- Loads have implicit target (register on top)
- Stores have implicit source (register on top)
- Arithmetic instructions have all operands implicit (n-ary operation takes n registers from top)

CPU registers

|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----| Flags |

memory

LOAD 2
LOAD 4
ADD
LOAD 1
LOAD 1
ADD
LOAD $F7
NOT
MUL
ADD
STORE x
• Load/store architecture
• Loads have implicit target (register on top)
• Stores have implicit source (register on top)
• Arithmetic instructions have all operands implicit (n-nary operation takes n registers from top)
Stack Machine

- Load/store architecture
- Loads have implicit target (register on top)
- Stores have implicit source (register on top)
- Arithmetic instructions have all operands implicit (n-nary operation takes n registers from top)

### CPU registers

- R15
- R14
- R13
- R12
- R11
- R10
- R9
- R8
- R7
- R6
- R5
- R4
- R3
- $F7
- 2
- 6
- R0

### Instructions

- LOAD 2
- LOAD 4
- ADD
- LOAD 1
- LOAD 1
- ADD
- LOAD $F7
- NOT
- MUL
- ADD
- STORE x

---

RegStackTop
Flags
(SP)
IP

memory
Stack Machine

- Load/store architecture
- Loads have implicit target (register on top)
- Stores have implicit source (register on top)
- Arithmetic instructions have all operands implicit (n-ary operation takes n registers from top)

CPU registers

|---|-----|----|----|----|----|----|-----|----|----|-----|----|----|-----|----|----|-----|----|----|-----|----|----|-----|----|----|-----|----|----|-----|----|----|-----|----|----|-----|----|----|-----|----|----|-----|----|----|-----|----|----|-----|

- LOAD 2
- LOAD 4
- ADD
- LOAD 1
- LOAD 1
- ADD
- LOAD $F7
- NOT
- MUL
- ADD
- STORE x

memory

RegStackTop
Flags
(SP)
IP

• Load/store architecture
• Loads have implicit target (register on top)
• Stores have implicit source (register on top)
• Arithmetic instructions have all operands implicit (n-ary operation takes n registers from top)
Stack Machine

- Load/store architecture
- Loads have implicit target (register on top)
- Stores have implicit source (register on top)
- Arithmetic instructions have all operands implicit (n-ary operation takes n registers from top)

CPU registers

<table>
<thead>
<tr>
<th>RegStackTop</th>
<th>Flags</th>
<th>(SP)</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

memory

- LOAD 2
- LOAD 4
- ADD
- LOAD 1
- LOAD 1
- ADD
- LOAD $F7
- NOT
- MUL
- ADD
- STORE x
Stack Machine

- Load/store architecture
- Loads have implicit target (register on top)
- Stores have implicit source (register on top)
- Arithmetic instructions have all operands implicit (n-ary operation takes n registers from top)
Stack Machine

- Load/store architecture
- Loads have implicit target (register on top)
- Stores have implicit source (register on top)
- Arithmetic instructions have all operands implicit (n-ary operation takes n registers from top)

CPU registers


RegStackTop
Flags
(SP)
IP

memory

LOAD 2
LOAD 4
ADD
LOAD 1
LOAD 1
ADD
LOAD $F7
NOT
MUL
ADD
STORE x
Stack Machine

- Load/store architecture
- Loads have implicit target (register on top)
- Stores have implicit source (register on top)
- Arithmetic instructions have all operands implicit (n-ary operation takes n registers from top)

### CPU registers

<table>
<thead>
<tr>
<th></th>
<th>R15</th>
<th>R14</th>
<th>R13</th>
<th>R12</th>
<th>R11</th>
<th>R10</th>
<th>R9</th>
<th>R8</th>
<th>R7</th>
</tr>
</thead>
</table>

- RegStackTop
- Flags
- (SP)
- IP

### Memory

- LOAD 2
- LOAD 4
- ADD
- LOAD 1
- LOAD 1
- ADD
- LOAD $F7
- NOT
- MUL
- ADD
- STORE x
Stack Machine

- Load/store architecture
- Loads have implicit target (register on top)
- Stores have implicit source (register on top)
- Arithmetic instructions have all operands implicit (n-ary operation takes n registers from top)

CPU registers

<table>
<thead>
<tr>
<th>?</th>
<th>R15</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>R14</td>
</tr>
<tr>
<td>?</td>
<td>R13</td>
</tr>
<tr>
<td>?</td>
<td>R12</td>
</tr>
<tr>
<td>?</td>
<td>R11</td>
</tr>
<tr>
<td>?</td>
<td>R10</td>
</tr>
<tr>
<td>?</td>
<td>R9</td>
</tr>
<tr>
<td>?</td>
<td>R8</td>
</tr>
<tr>
<td>?</td>
<td>R7</td>
</tr>
<tr>
<td>?</td>
<td>R6</td>
</tr>
<tr>
<td>?</td>
<td>R5</td>
</tr>
<tr>
<td>?</td>
<td>R4</td>
</tr>
<tr>
<td>?</td>
<td>R3</td>
</tr>
<tr>
<td>8</td>
<td>R2</td>
</tr>
<tr>
<td>16</td>
<td>R1</td>
</tr>
<tr>
<td>22</td>
<td>R0</td>
</tr>
</tbody>
</table>

global data: ...

memory

LOAD 2
LOAD 4
ADD
LOAD 1
LOAD 1
ADD
LOAD $F7
NOT
MUL
ADD
STORE x
Stack Machine

- Load/store architecture
- Loads have implicit target (register on top)
- Stores have implicit source (register on top)
- Arithmetic instructions have all operands implicit (n-ary operation takes n registers from top)

```
LOAD 2
LOAD 4
ADD
LOAD 1
LOAD 1
ADD
LOAD $F7
NOT
MUL
ADD
STORE x
```

CPU registers

```
<table>
<thead>
<tr>
<th></th>
<th>R15</th>
<th>R14</th>
<th>R13</th>
<th>R12</th>
<th>R11</th>
<th>R10</th>
<th>R9</th>
<th>R8</th>
<th>R7</th>
<th>R6</th>
<th>R5</th>
<th>R4</th>
<th>R3</th>
<th>R2</th>
<th>R1</th>
<th>R0</th>
</tr>
</thead>
</table>
```

- RegStackTop
- Flags (SP)
- IP

Memory

Global data:

```
... ...
... ...
... ...
... ...
... ...
```

Address: 22

- Memory
How To Write/Compile Complex Expressions for Stack Machine?

\[ x := 1 + a + F_1(b + c) \times F_2(d + e + F_3(2, 3), f + F_4(4) + g) + F_5(5) + x; \]
x := 1 + a + F1(b + c) * F2(d + e + F3(2, 3), f + F4(4) + g) + F5(5) + x;

- Instruction reference:
  ldc.i4.X = load constant value X as 4 byte integer = immediate
  ldsfld a = load from static field a = load from global variable a
  stsfld x = store to static field x = store to global variable x

- Arguments to functions passed on top of register stack
- Return values from functions stored on top of register stack
Complex Expressions in Stack Machine’s Assembler (.NET CIL Code Here)

\[
x := 1 + a + F1(b + c) * F2(d + e + F3(2, 3), f + F4(4) + g) + F5(5) + x;
\]

- Instruction reference:
  - `ldc.i4.X` = load constant value \(X\) as 4 byte integer = immediate
  - `ldsfld a` = load from static field \(a\) = load from global variable \(a\)
  - `stsfld x` = store to static field \(x\) = store to global variable \(x\)

- Arguments to functions passed on top of register stack
- Return values from functions stored on top of register stack

Pros (advantages):
- Easy to write by humans
- Easy to read by cross compilers
  → implemented by virtual machines (.NET’s CLR, JVM)
  → allows easier implementation of JIT compiler from IL code (CIL / Java bytecode) to target CPU’s machine code (e.g. x86)

Cons (disadvantages):
- Hard to generate effective code for modern CPUs (modern compilers are be able to generate more effective code for general register architecture CPUs)
  → not implemented by typical real CPUs