Lecture 22
Part 1
Overall Plan

- Deliver lecture content ahead of regular lecture time
- During regular lecture time: synchronous Q&A via Piazza/Zoom
  - Zoom link: https://buffalo.zoom.us/j/225259023
- After regular lecture time: asynchronous Q&A via Piazza
Lecture content

- Slides - I am adding more slides/details to make them easier to follow when reading.

- Will try different ways of providing supporting materials: written content and recorded voice with slides.

- Recordings will be kept short, so one lecture may have several recordings associated with it.

- Share your thoughts:
  - anonymously in CSE443: What's on your mind GoogleForm
  - privately or publicly in CSE443 Piazza post
Team meetings

- If we can run them synchronously at regular times, great!
- We will try this week, and see how it goes.
- It is VERY IMPORTANT that all team members stay engaged!
Homeworks

- HW01 grading to be done shortly - grade via UBLearns gradebook. Will post general feedback to Piazza.
- HW02 grading to be done immediately after HW01 - grade via UBLearns gradebook. Will post general feedback to Piazza.
- HW03 is posted - note due date is different from PR03 due date.
Projects

- PR01 was autograded - all teams scored 100%.
- PR02 grading will be started soon - if all goes smoothly PR02 grades will be available within about a week.
- PR03 is posted - note due date is different from HW03 due date.
Questions?

If you have questions, please post them in Piazza post @107 "LECTURE 22 (3/23/2020) Q&A post"
CSE4433 Compilers

Lecture 22

Part 2
Review

- Project 1: char stream $\rightarrow$ LEXER $\rightarrow$ token stream
- Project 2: PARSER builds symbol table, checks for undefined or multiply defined names from token stream.
- Project 3: PARSER will also perform type checking and generate intermediate code.
Intermediate Representation (IR)

we'll use IR defined in textbook: the "three address code"
Three address code instructions
(see 6.2.1, pages 364-5)

1. \( x = y \text{ op } z \)
2. \( x = \text{ op } y \)  
   (treat \( i2r \) and \( r2i \) as unary ops)
3. \( x = y \)
4. goto \( L \)
5. if \( x \) goto \( L \) / ifFalse \( x \) goto \( L \)
6. if \( x \) relop \( y \) goto \( L \)
7. function calls:
   - param \( x \)
   - call \( p, n \)
   - \( y = \text{ call } p \)
   - return \( y \)
8. \( x = y[i] \) and \( x[i] = y \)
9. \( x = \&y, x = \star y, \star x = y \)
Three address code instructions
(see 6.2.1, pages 364-5)

1. \( x = y \text{ op } z \)
2. \( x = \text{ op } y \) (treat \( i2r \) and \( r2i \) as unary ops)
3. \( x = y \)
4. goto L
5. if \( x \) goto L / ifFalse \( x \) goto L
6. if \( x \) relop \( y \) goto L
7. function calls:
   - param \( x \)
   - call \( p, n \)
   - \( y = \text{ call } p \)
   - return \( y \)
8. \( x = y[i] \) and \( x[i] = y \)
9. \( x = \&y, x = *y, *x = y \)

We’ll start with these.
We’ll spend significant time on function calls later.
We’ll explore these as needed later on.
type information

- What information does a type convey?

- How is type information used during compilation?
type information

What information does a type convey?
- type indicates size
- type indicates storage location
  (a) primitives: either stack or heap
  (b) records: on heap (via pointer)
  (c) arrays: on heap (via pointer)
  (d) functions: code in static, locals on stack

How is type information used during compilation?
type information

What information does a type convey?
- type indicates size
- type indicates storage location
  (a) primitives: either stack or heap
  (b) records: on heap (via pointer)
  (c) arrays: on heap (via pointer)
  (d) functions: code in static, locals on stack

How is type information used during compilation?
- determines how to lay out records, arrays, invocation records in memory
- determines how to translate names in program to memory accesses
- determines which instructions to use to manipulate values in memory
Sizes of types

- int: 32 bits (2's complement)
- real: 64 bits (IEEE 754)
- Boolean: 8 bits (TBD)
- character: 8 bit (ASCII)
Sizes/layouts of values of types

- **type string**: 1 → character  
- 4 bytes + length of string * size of character (= 1 byte)
- # of dimensions is part of type

<table>
<thead>
<tr>
<th>size of dimension 1 (integer)</th>
<th>(0)</th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
<th>(4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 S</td>
<td>V</td>
<td>A</td>
<td>X</td>
<td>E</td>
<td>S</td>
</tr>
</tbody>
</table>

Array layout in memory

Two options:

- row-major
- column-major

Textbook discusses on page 382; row-major and column-major refer to two-dimensional arrays, but can be generalized for arrays with more dimensions.
Row-major array layout

What is the size of an $X$-dimensional array of type $T$?

sizes of dimensions $(S_i)$: $X \times 4$ bytes

data: $(T \times S_i) \times \text{sizeof}(T)$

Example shows two-dimensional array (2 rows, 3 columns)

<table>
<thead>
<tr>
<th>size of first dimension</th>
<th>size of second dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>first row</th>
<th>second row</th>
</tr>
</thead>
<tbody>
<tr>
<td>a(0,0)</td>
<td>a(1,0)</td>
</tr>
<tr>
<td>a(0,1)</td>
<td>a(1,1)</td>
</tr>
<tr>
<td>a(0,2)</td>
<td>a(1,2)</td>
</tr>
</tbody>
</table>
Column-major array layout

What is the size of an $X$-dimensional array of type $T$?

sizes of dimensions ($S_i$): $X \times 4$ bytes

data: $(\prod_{i \in X} S_i) \times \text{sizeOf}(T)$

Example shows two-dimensional array (2 rows, 3 columns)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>size of first dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>size of second dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>a(0,0)</td>
<td></td>
<td>first col</td>
</tr>
<tr>
<td>a(1,0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a(0,1)</td>
<td></td>
<td>second col</td>
</tr>
<tr>
<td>a(1,1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a(0,2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a(1,2)</td>
<td></td>
<td>third col</td>
</tr>
</tbody>
</table>
Propagation of type information in parse tree
For the purposes of type checking the number of dimensions is relevant, but the size of each dimension is not.
What if type info comes after dimensions?

Figure 6.16, p 375

\[ w = 2 \times 3 \]
\[ w = 6 \]
\[ w = 9 + 6 \times 1 = 15 \]

9+array(2,arr(3,character))

© 2020 Carl Alphonce - Reproduction of this material is prohibited without the author's consent
Planting a seed...

Q: if $a$ and $b$ are compatible array types, what are the semantics of $a := b$?
Planting a seed...

Q: if a and b are compatible array types, what are the semantics of a := b?

A: We won't answer this now, but think about it...