2022年4月24日 22:52

WEEK 0

Feb 19 (Right After Midterm)

Organising the structure of the compiler.

- Stack
- Lexer Parser

WEEK 1

Feb20-26

- **Goal:**

1. **Have a general view on the overall structure of compiler.** (Context, Paser, Lexer …) *(5 hours)*
2. **Have a clear view on StackFrame design.** This is marked as a milestone as it is not included in the previous labs. Many instructions are context related hence having stackframe taken in early consider would make the progress go more smoothly) *(7 hours)*
3. **Implementation of Parser and Lexer.** *(7 hours)*

2/21 (George && Jeffrey)

Function of the stack frame pointer.

Start to do research on the stack frame

2/23 (George && Jeffrey)

Size of the stack frame

Tried exploring potential rules about the stack frame size according to the Godbolt. How it related types, whether or not function call is within the scope… so on.

We head into this approach bc don't know where to start.

(Update #date#) **wrong decision had been made here**

(What would be a proper direction?), the stack frame size varies depends on tons of others factors : type…)

2/25 (George && Jeffrey)

An idea formed and according to the rules we observed from Godbolt.

Investigate on C(90) standard parser and lexer.

2/26 (George && Jeffrey)

Built the parser and lexer.

We disable most of the parser part and only keep simplest addition for simplicity.

Goal Completed status:

Completed:

- Implemented the Parser and Lexer within expected time. (Both)
- Formed a basic idea on designing the structure of the Stack Frame.

Generally meet our expectation

Time consumption: Average 5 hours per day

WEEK 2

Feb27-Mar5

- **Goal:**

1. **Implementation of Context (10 hours)**
2. **Base Class for AST (declaration component statement) (10 hours)**

    * a file containing just a single function with no arguments
    * variables of `int` type
    * local variables

3. **Generate some basic MIPS code. (10 hours)**
4. **Implementation of INT type and local variables. (5 hours)**
5. **Configure the structure of the stack and decide how the stack and context are related and used during compiling.**

2/27 (George && Jeffrey)

Base on the information collected from last week we started to **write the Contex**t and **Node part (Visitor Design).**

Implementation of Node Part

We want the following classes related to different Parser's layers to inherit from the general Node.

Implementation of Declaration and Component statement. (Half completed)

Implementation of Context

We decided to **build and pass** the context during the GeneratingMips process.

The other method would be building up the context alone side with the formation of AST tree.

We couldn't tell the difference at this point. The point we have considered by far is that both method required to go through the tree at least twice. Hence, there shouldn't be a much difference in term of performance.

(Update #date#) wrong decision had been made here

it significantly complicates our programme in the following part.

2/28 (George && Jeffrey)

At this point, we are aiming for implementation of the framework first, hence lots of details had being missing.

Notice more situations in the size of the function stack.

For example, if a variable is declared but not be called, a function called inside a function would add 16 to the function stack size, etc.

At this point we realized there might be more implicited rules for specifying the stack size, we didn't dig inside as we decided to finish the basic function first and all these rules should be enough.

3/1-3/2 (George && Jeffrey)

Stopped compiler because of the other course work. (Inform processing)

3/3 (George && Jeffrey)

Finished up building the context and Node part.

Integrating context formation into Node part.

Leave space for the double and char type.

Implement a simple function, choose to build the context within the function.

3/4 (George && Jeffrey)

Implementation of addtion arithmetic operation.

Successfully generating mips for simple functions with only addition inside.

Goal Completed status:

Completed:

- Implemented a basic context, successfully generate simple addition code.

Generally meet our expectation

Time consumption: Average 5 hours per day

WEEK 3

Mar6-12

- **Goal**

1. **Implementation of all the int type 3 operand arthmetic operation (3-4 hours)**

   * arithmetic and logical expressions

2. **Implementation of control flow (3-4 hours)**

   * if-then-else statements
   * while loops

3. **Implementation of Functions (15 hours)**

   * Function arguments (including situation where there are more than 4 arguments)

   *Multiple functions that call each other.

   *recursive function call

   *calling externally-defined functions

4. **Add extra dynamic context to solve the problem of using different variables at the same time. (8-9 hours)**

3/6 (George && Jeffrey)

Finish the local int variable declaration and call. Follow the dynamic stack size.

- Dynamic stack: idea form previously. Used for arithmetic calculation. Size is deduced during formation of AST tree.
    - At this point we are taking "efficiency" into account. We believed all the information could be retrieve during the formation of the AST tree.

(Update #date#) **wrong decision had been made here**

- Efficiency: This is way more complicated than what we thought. The idea of deducing dynamic stack size during the formation of tree couldn't cover cases of Struct and so on… (information at that point is not completed i.e. the size of the type remain unknow till certain point.

- which complicated the following code and structure and limited the design decision we could made.

Finish all the basic binary and unary arithmetic operation parts.

3/7 (George && Jeffrey)

Build If-then-else statements. While loops.

Finish the basic features part.

3/8 (George && Jeffrey)

Start to build Functions but only finished part of it.

3/9-3/10 (George && Jeffrey)

Stopped doing coursework and moved to information processing courework.

3/11-3/12

Finnish Function part (George)

Do some simple testing. Start to write dynamic context to temporarily store the register value. This is used to avoid covering useful values in the registers during multi variable calculation. (Jeffrey)

1st Milestone

Completed all the basic features in the requirement.

Goal Completion status:

Completed:

Implemented a basic context, successfully generate simple addition code.

Generally meet our expectation

Time consumption: Average 7hours per day

WEEK 4

Mar13-19

- **Goal:**

1. **Implementation of Array type (globally and locally) (10 hours)**

   *Array declaration (globally and locally)

   *Reading and writing elements of an array

2. **Implementation of Enumeration type (4 hours)**
3. **Implementation of Break and Continue (2 hours)**
4. **Implementation of Switch (3 hours)**

5. **Start to do global variable declaration (8 hours)**

3/13-3/14 (George && Jeffrey)

Completing the array part implementation.

3/15 (George && Jeffrey)

Discover the situation int x, y, z and consider all the possible situation for ‘,’ structure; Reconfigure the Compound part parser. (We spent a long time on this > 8 hours to figure them out)

3/16 (George && Jeffrey)

Finish switch, break and continue.

Almost finished the Enumeration part

3/18-3/20 (George and Jeffrey)

Finished Enumeration part.

Start to build the testing part, including the makefile.

Find many bugs. We spend the whole weekend fixing them.

2nd Milestone

Completed all the intermediate features in the requirement.

Goal Completion status:

Completed:

Finished all the intermediated features and figure out all the bugs related to this. According to the test our compiler could pass all the related programs.

Not Completed:

Since we spent a long time fixing bugs and doing tests, we do not have any extra time to start doing global variable declarations.

Partly meet our expectation, not very satisfied

Time consumption: Average 8hours per day

WEEK 5

Mar20-26

Goal:

1. **Implementation of the Struct type (5 hours)**
2. **Implementation of the Pointer type (including pointer to local and global type) (10 hours)**

   *Taking the address of the variable

   *Dereferencing

   *Pointer arithmetic

3. **Implementation of the Double and Float type (15 hours)**

3/21 (George && Jeffrey)

Finished doing Struct type.

3/22 - 3/24

Since we have Information Processing deadline, Communication Lab Oral and Control Lab Report deadline during this week, we spent more time on this.

3/25 (George && Jeffrey)

Finish Pointer for local variables.

3/26-3/27 (George && Jeffrey)

Double type and Float type

3nd Milestone

Completed Double and Float types which requires extra new configuration to generates code.

Goal Completion status:

Completed:

Almost completed all of our goals. A productive week.

Not Completed:

Some parts of float type implementation are not completed due to time limitation, but since we build double type configuration successfully, we do not spend quite a long time finishing this part.

The Process is hard especially regarding the double and float type. But we managed to finish this.

Time consumption: more than 8 hours on average per day

WEEK 6

Mar28-29

Goal:

Implementation of all types of variables (+ 10 hours)

Implementation of 'sizeof' function (1hour)

Implementation of escape sequences like '\n' (1 hour)

Implementation of 'typedef' (3hours)

Implementation of pointer to double type array (4 hours)

3/28 (George && Jeffrey)

Char, String and unsigned type type

Global pointer, global array etc.

3/29 (George && Jeffrey)

Fixing error met during test.

Finished 'sizeof' function.

Try to implement pointer to double type array.

Try to implement 'typedef'

Final Milestone

Completed 80% of Advanced Requirement

Goal Completion status:

Completed:

Except typedef implementation, we almost finished everything. However, unfortunately, when we are trying to configure the pointer to double type array, we change somewhere by mistake, which makes our compiler cannot compile some parts of double type arithmetic programs.

Not Completed:

Typedef, double type arithmetic goes wrong.

Partly meet our expectation, not very satisfied

Time consumption: Average 24 hours per day

Addition implementation

Declaration of variable, array, function of the same type within one code.

Double pointer.