令和 5 年 2 月 13 日

京都工芸繊維大学コンピュータ部

Lime 61

はじめに

コンピュータ部部長の寺村です。この度はお忙しい中、我々コンピュータ部にご興味、ご関心を持っていただきありがとうございます。そして、この Lime62 号を手にしてくださったことを、部を代表いたしまして厚く御礼申し上げます。この Lime は部員の活動の一部を記したものです。日々の活動の成果や部員個人の趣味など、内容は多岐にわたります。これを通じて、コンピュータ部のことを知っていただき、少しでも我々の活動に興味を持っていただければ幸いです。そして、今回の Lime 作成にあたり編集を担当した木村君、記事を執筆した部員各位、ならびに OB・OG の方々を含め、日々我々の活動を支えてくださっているすべての方々への感謝をもって始めの挨拶とさせていただきます。

令和 5 年 2 月 13 日

京都工芸繊維大学コンピュータ部部長　寺村 英之

目次

1 ビンゴゲーム

機械工学課程 1 回生 絹川 和

1.1 はじめに

HTML, CSS, JavaScript でビンゴゲームを作ろうと思いました。まだ途中で完成していないのですが、最低限はできたので書かせてもらいます。これは、ドットインストールというサイトにあったビンゴシート作成のコード (https://dotinstall.com/lessons/bingo_js) を軸に作っています。このサイトのコードはサイトを開いたときに５×５のビンゴシートを作成するもので、また行ごとに入る数字が決まっています。その辺の修正ともう少しビンゴをサイト上で扱えるようにしました。

1.2 作れたビンゴゲーム

- ビンゴの数範囲の指定
- ビンゴの大きさ指定
- ランダムでビンゴ作成
- 一回目の答え表示（答えはビンゴの数範囲の数字全部からランダムに表示される。二回目からは答えがバグりました）
- クリックすることで色を変える（数字が一致していなくてもクリックで色が変わります）

作れたビンゴゲームはこんな感じでだいぶ穴だらけです。下が作った物の写真とコードです。

図 1.1 この図を出力する記述の一例

```
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="utf-8">
  <title>Bingo</title>
  <link rel="stylesheet" href="css/styles.css">
</head>
<body>
  <h1>CREATE BINGO</h1>
  <div class="dif">
    <label>数字の範囲：<input type="text" id="range" value="100"></label></br>
    <label>行列の数 ：<input type="text" id="length" value="5"></label>
    <button id="el">決定</button>
  </div>
  <div id="text"></div>
  <div id="answerScreen"></div>
  <table>
    <tbody>
    </tbody>
  </table>

  <script src="js/main1.js"></script>
</body>
</html>
```

```
'use strict';

{
  /* BINGO生成 */
  document.getElementById('el').addEventListener('click',()=>{
    /* 定数の生成 */
```

```
7       // 範囲指定の数字を受け取る
8       const number = document.getElementById('range');
9       const range = number.value;
10
11      // 行列の数を受け取る
12      const row = document.getElementById('length');
13      const length = row.value;
14
15      /* ビンゴ生成準備 */
16      function setUp(){
17        // テキストの削除
18        let text = document.getElementById('text');
19        text.textContent = '';
20        // ビンゴの削除
21        let bigngo = document.querySelector('tbody');
22        bigngo.remove();
23        // ビンゴ生成の準備
24        const tbody = document.createElement('tbody');
25        document.querySelector('table').appendChild(tbody);
26
27      }
28
29      /* ビンゴ生成可能か判断 */
30      function judge(){
31        // 数字でないとき、ゼロより数字が大きい時
32        if(range <= 0 || length <=0 || isNaN(range) || isNaN(length)){
33          text.textContent = "自然数を入力してください";
34          return 0;
35        }
36        // 数字の範囲が行の２乗より小さい時
37        if(range < length**2){
38          text.textContent = "数字の範囲が行の２乗より小さいのでビンゴを作成できません。
                ";
39          return 0;
40        }
41      }
42
43
44      function createBingo(){
45        const source = [];
46        for(let i = 0; i < range; i++){
47          source[i] = i + 1;
48        }
49        const columns = [];
50        for(let i = 0; i < length; i++){
51          columns[i] = [];
52          for(let j = 0; j < length;j++){
53            columns[i][j] = source.splice(Math.floor(Math.random()*
                  source.length),1)[0];
54          }
```

```
55          }
56          if (length % 2 ===1){
57            const center = Math.floor(length / 2);
58            columns[center][center] = 'BINGO';
59          }
60          return columns;
61        }
62
63
64        function renderBingo(columns){
65          for(let row = 0; row < length; row++){
66            const tr = document.createElement('tr');
67            for (let col = 0; col < length; col++){
68              const td = document.createElement('td');
69              td.textContent = columns[col][row];
70              tr.appendChild(td);
71            }
72            document.querySelector('tbody').appendChild(tr);
73          }
74        }
75        setUp();
76        judge();
77        const columns = createBingo();
78        renderBingo(columns);
79
80        /* ビンゴ開始 */
81        //必要な定数
82        const td = document.querySelectorAll('td');
83        const tdL = td.length;
84        if (length % 2 ===1){
85          const center = Math.floor(length ** 2 / 2);
86          td[center].classList.add('clicked');
87        }
88        //配列をランダムに並び替え
89        function fisherYatesShuffle(arr){
90          for(let i =arr.length-1 ; i>0 ;i--){
91              let j = Math.floor( Math.random() * (i + 1) ); //random index
92              // [arr[i],arr[j]]=[arr[j],arr[i]]; // swap
93              let tmp = arr[i];
94              arr[i] = arr[j];
95              arr[j] = tmp;
96          }
97          return answers;
98        }
99        //答えの数字をセット
100       const answers = [];
101       let answerScreen = document.getElementById('answerScreen');
102       function setAnswers(){
103         for (let i = 0; i < range; i++){
104           answers[i] = i+1;
```

```
105         }
106         fisherYatesShuffle(answers);
107         return answers;
108       }
109       let count = 0;
110       function displayScreen(){
111         answerScreen.textContent = answers[count];
112         count++;
113       }
114       setAnswers();
115       displayScreen();
116       setInterval(displayScreen,5000);
117       console.log(answers);
118
119       /* クリックしたら色変わる */
120       function click(){
121         for(let i = 0; i < td.length; i ++){
122           td[i].addEventListener('click',function(){
123             td[i].classList.toggle('clicked');
124           });
125         }
126       }
127       click();
128     });
129   }
```

```
1   body{
2     font-family: 'Courier New', monospace;
3     font-family: bold;
4     background: orange;
5     color: white;
6   }
7
8   h1{
9     display: flex;
10    justify-content: center;
11    color: white;
12  }
13
14  .dif{
15    margin: 0 auto;
16    position: relative;
17    background: #000;
18    border-radius: 10px;
19    width: 300px;
20    height: 125px;
21    padding: 20px;
22  }
23
24  label{
```

```
25    display: block;
26    padding: 0;
27    margin: 0;
28  }
29
30  #el{
31    border-radius: 50%;
32    width: 50px;
33    height: 50px;
34    position: absolute;
35    bottom: 10px;
36    left: 150px;
37    cursor: pointer;
38  }
39
40  #text{
41    display: block;
42    display: flex;
43    justify-content: center;
44    font-size: 30px;
45    padding: 10px 10px;;
46  }
47
48  input{
49    border-radius: 5px;
50    padding: 1px;
51  }
52
53  #answerScreen{
54    width: 100px;
55    height: 100px;
56    background: #000;
57    float: left;
58    color: white;
59    font-size: 40px;
60    font-weight: bold;
61    text-align: center;
62    vertical-align: middle;
63  }
64
65  table{
66    margin-top: 30px;
67    padding: 30px 30px;
68    margin: 0 auto;
69  }
70
71  td{
72    background: white;
73    width: 60px;
74    height: 60px;
```

```
75    text-align: center;
76    line-height: 40px;
77    color:orange;
78    cursor: pointer;
79    user-select: none;
80  }
81
82  td.clicked{
83    background: #000;
84  }
```

1.3 大変だったこと

HTML, CSS, JavaScript の勉強を始めて一ヶ月たっていないのでまだどう書けばどう動くのかよく分かりませんでした。また、id や class、変数や定数、などの名前の付け方の自分ルールがないのでどれがどれだか分からなくなったりして大変でした。後はエラー表示されてればどこがダメなのかがすぐに分かるけれどエラー表示がないときはどこがよくないのか探すのに時間がかかりました。

1.4 今の課題

今回作ったビンゴは答えがビンゴの数範囲の数字全部からランダムに表示されるもので、二回目からは答えがバグります。consol.log で配列などを確認した限り作られた配列に間違いはなかったので、一回目に作った定数とかが初期化されていないのかもしれないなと思いました。けど、直し方も全く思いつかないのでまったり考えようと思います。あと、クリックするのが答えと違っていても色が変わってしまうのもどうにかしたいです。これは答えがバグるのをどうにかできれば for と if 文でなんとかできると思っています。最後にスコアを出せるようにしたいですが、行は行数ごとに色が変わっているかを見て判断するのと、列はも行数跳びの数が色が変わっているのかを判断することができればビンゴになっているかは分かると思います。

1.5 最後に

HTML, CSS, JavaScript はとても学びやすい言語ではあるなと思いました。これらは、ブラウザ上で簡単に表示することができ、ある程度なら簡単に動きをつけることができます。自分は飽き性なので、このようなすぐに結果が見える言語は勉強がしやすくとて

もありがたいなと思いました。今回まだ完成はしていないけれどある程度一つのものを形にすることができたのでとてもうれしかったです。また、分からないながらも何かを作ろうとすれば、ただただ勉強動画を見ているよりもいろんなことを吸収できた気がします。

しかし、Web 作成では使わなければいけない言語は最低でもこの 3 語であと、PHP などのサーバー側で動かす言語もやらなければいけないとのことでなかなか勉強できるか怪しいなとも思ってしまっています。特に、PHP はサーバー側の動きらしく HTML, CSS, JavaScript のように見た目にダイレクトに影響を与える言語ではないそうなので勉強できるか不安ですが、データ処理ができれば作れる物の幅が大きく上がると思うのでこれからもこれらの言語の勉強も続けていきたいなと思いました。

あと、Arduino のスターターキットを買って電子工作も始めたのですがこちらはさっぱりです。コンデンサーの使い方など全く分かりません。しかし、電子工作ができるようになれば、PC 上だけでなくリアルのほうでもいろんなことができるようになると思うのでこれの勉強もしっかりしたいなと思いました。だいぶ話がそれてしまいましたが最後まで読んでいただきありがとうございました。

2 Markdownによるスライド作成

情報工学課程 2 回生 木村 俊星

2.1 本稿について

コンピュータ部では，毎週「オンライン Python 講座」が開かれている．当講座内で山上さんが共有および配布してくださる資料には，LaTeX 風の数式が登場したり，ソースコードがそのままの形式でベタっと張り付けてあったり，Powerpoint で作成されたとは思えないものである．何を使って資料を作成しているのか，山上さんに尋ねたことはあるが，詳細は忘れてしまった．とにかく，

- Markdown 形式で書かれたコードを基にスライドができていること
- vscode の拡張機能が使われていること

だけは覚えていた。これらの断片的な情報を頼りに，自らスライドを作成してみるまでに得られた知識の一部をまとめたものが本稿である．ただし，Markdown 記法自体についての解説は行っていない．読者自身で検索していただいた方が恐らく理解は早い．

Markdown 形式でスライドを作成する方法はいくつか存在するようである．利用できる vscode の拡張機能として以下の 2 つが見つかった．

- Marp for VS Code
- vscode-reveal(以下，reveal)

今回はこの両方を使い比べてみようと思う．

2.2 Marp for VS Code

冒頭で Marp の有効化と，スライドの書式設定を記述する．`style` で定義したスタイルは，任意のスライドの冒頭で`<!-- class: hoge -->`のように指定できる．

例えば，図 2.1 で定義している title を適用したい場合は，<!-- class: title -->とスライドの冒頭に記述する．

```
---
marp: true
theme: default <!-- {default(=white), gaia(=cream)} -->
paginate: true <!-- ページ番号 -->
header: ""
footer: ""
size: 16:9     <!-- {4:3, 16:9, A0-A8, B0-B8} -->
style: |
    section.title {
        justify-content: center;
        text-align: center;
    }

    section {
        justify-content: start
    }
---
```

図 2.1 Marp のスライド設定

ここに記述した style 以外の設定は全スライドに適用される．特定のページのみ異なる設定にしたい場合は，そのスライドの冒頭に図 2.2 のように記述する．ディレクティブ名の先頭にアンダースコア (_) をつけることで，現在のスライドにのみ設定が適用される．アンダースコアをつけなかった場合，そのスライド以降の全てのスライドに設定が適用される．

```
<!--
_color: white
_footer: Photo by xxx
-->
```

図 2.2 現在のスライドにのみ適用

ページの区切り位置はハイフンを 3 つ並べて指定する (---)．あとは，Markdown 記法に則ってページ毎に内容を記述していけばよい．実際に作成したみたスライドを図 2.3 に示す．対応するコードを掲載すると，それだけで 1 ページ消費してしまうので割愛することにした．

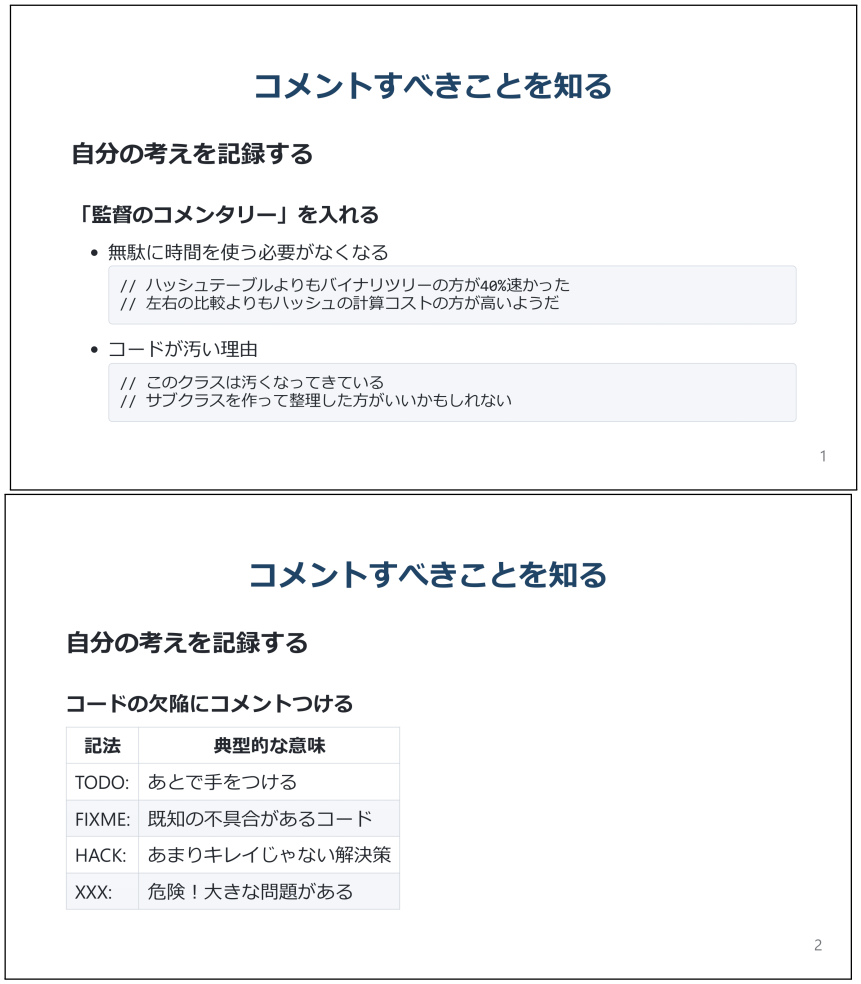

図 2.3 Marp による試作スライド

2.3 vscode-reveal

冒頭でスライドの書式設定を行う点は Marp と同様であるが，reveal ではスライドの切り替え方 (`transition`) が指定できる．

```
1 ---
2 theme: "black"     <!-- {black, white, simple,...} -->
3 transiton: "slide" <!-- {fade, slide, zoom, ...} -->
4 slideNumber: true  <!-- ページ番号 -->
5 title: what to comment
6 ---
```

図 2.4 reveal のスライド設定

ページの区切り位置も同じくハイフンを 3 つ並べて指定するが，ハイフンを 2 つ並べると，スライドを横ではなく縦に切り替えることができる．スライドを横一列に並べると内容の区切りがつきにくいが，同じ話題の内容を縦に並べ，話題が切り替わる時点で横に切り替えるようにすれば，スライドの構造が分かりやすくなる．

実際に作成してみたスライドの例を図 2.5 に示す．

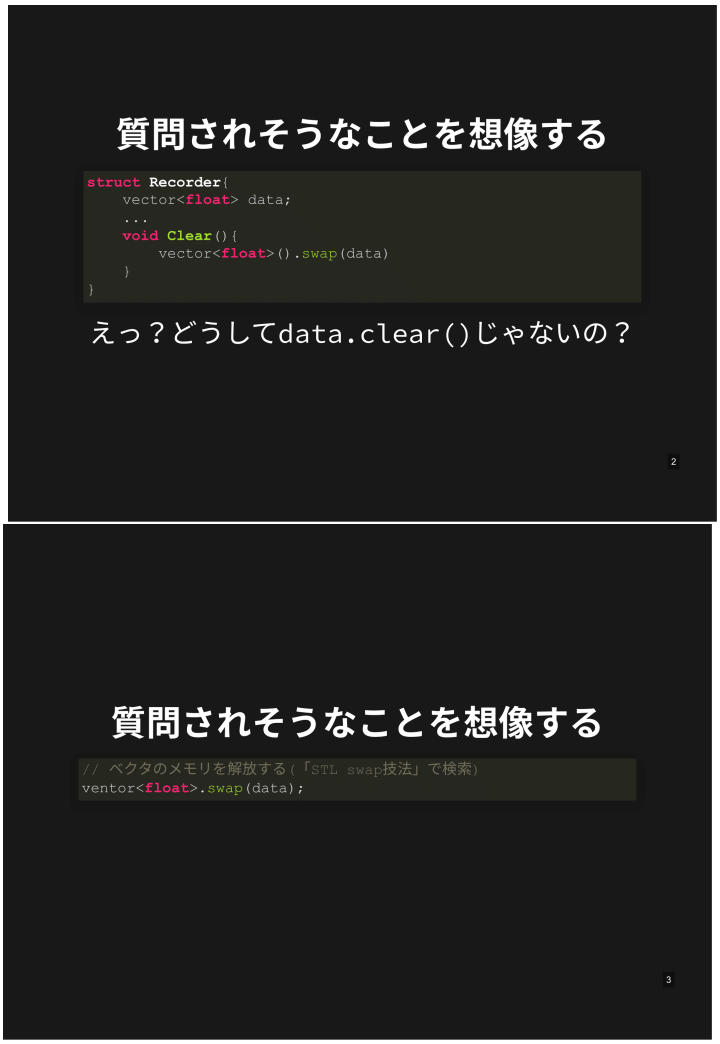

図 2.5　reveal による試作スライド

2.4 最後に

Powerpoint でスライドを作成するとなると，全くの白紙からデザインを考えないといけない．柔軟性が高いというのはメリットであるが，その分ユーザが考えなければならないことは多くなる．今回のように，Markdown 記法でスライドの構成を記述していく方法は，柔軟性の点では Powerpoint に劣るが，コンポーネントの体裁や配置の調整に時間をかけなくて済むので気楽だ．1 ページあたり数行のコードを書けば，ある程度体裁の整ったスライドを作成できるというのが，最大のメリットだと思う．他にも，

- Powerpoint ではできない，もしくはやろうとするとそれなりにコストがかかることを手軽に実現できる．e.g. ソースコード，数式，背景画像
- テキストベースなので編集が容易．
- 既存の Markdown ファイルをスライドに変換できる．

などのメリットが存在する．ただ，細かい装飾や配置，アニメーションに拘りたいときは，Markdown で実現しようとすると余計に時間がかかるので，やはり Powerpoint で作成する方がよい．

参考文献

[1] https://qiita.com/takeshisakuma/items/5a61e6eac123d28602fb
[2] https://qiita.com/tomo_makes/items/aafae4021986553ae1d8
[3] https://qiita.com/pocket8137/items/27ede821e59c12a1b222
[4] https://tracpath.com/works/development/marp/#i-7
[5] https://qiita.com/myname6c7c2/items/1d8c434f14dfe8f273a6
[6] https://qiita.com/Targityen/items/40ae4795e2cb77c1adc6
[7] https://daiblog923.com/vscode-revealjs/#index_id7

3 コンパイラを作ろう その1

情報工学課程 2 回生 中園 康聖

低レイヤの面白さに気づいたのはいいが、時間が足りなかった . . . やっぱテスト週間は大変だね。

3.1 CPU とメモリ

コンピュータを構成する要素は大きく分けると CPU とメモリになる。CPU はコンピュータにとって脳のようなものでデータを計算したりする場所である。それに対し、メモリは大きな配列のようなもので、データを格納することを主目的としている。コンピュータのデータのほとんどはメモリに格納されている。しかし、メモリだけが記憶領域というわけではない。CPU の中にも記憶領域がある。この記憶領域のことを**レジスタ**という。メモリは、CPU から見ると外部装置なので、アクセスするのに時間がかかる。できればそんな遅延は発生してほしくない。そこでレジスタにデータを入れることでその遅延の発生する回数を減らしている。あなたが図書館に行って調べ物をしているときをイメージしてくれるとわかりやすいと思う。あなたは物事を調べるときにわざわざ本棚に行って本を開いてデータを知った後、その本を元の場所に戻して、自分が作業していたテーブルに戻るだろうか。その本をテーブルに持ってきて、作業が終わるまでそこにおいて次調べるときに楽になるようにしているのではないだろうか。CPU の設計でもこんなことが考えられていてレジスタが作られている。レジスタにメモリで読み取った値を入れて、それを作業で使っている。CPU を操作するプログラムである機械語は、基本的に 2 つのレジスタ間で演算を行うようにできている。よって、基本的な CPU の作業は

- メモリから値を読み出し、レジスタに格納する
- 2 つのレジスタの間で演算を行う
- 演算結果をメモリに書き戻す

となっている。

3.2 アセンブラ

機械語はあくまで CPU が読むための物なので、人間にとっては非常に読みづらい。これを開発しろというのは至難の業である。そこで出てきたのが**アセンブラ**である。アセンブリは機械語とほぼ 1 対 1 の関係だが、人間にとっては圧倒的に読みやすい。コンパイラのほとんどはこのアセンブリを出力している。一見、機械語を出力しているようなコンパイラでも、実際はバックでアセンブラが稼働しているということも多い。アセンブリを機械語に出力することを「コンパイルする」というが、アセンブリであることに注目して「アセンブルする」ということもある。

objdump 命令を使うと実行ファイルから逆アセンブルができる。例えば”objdump -d -M intel /bin/ls”というコマンドを使うと ls 命令のアセンブリコードが読める。これを使ってアセンブリコードを確認してみよう。

objdump に内蔵されているデータは基本的に、

- 機械語のアドレス
- 実際の機械語の中身
- 対応するアセンブリの中身

という感じで順に入っている。

objdump のほかにアセンブリの勉強で役に立つオンラインサイトがある。それは Compiler Explorer(https://godbolt.org/) である。通称 godbolt と呼ばれるこのサイトは、左半分にコードのテキストを入力し、そのコードに対応するアセンブラが右半分に出力されるという構図になっている。コードを書き換えてすぐ結果が見れるので、ぜひ使ってみてほしい。

3.3 整数 1 個をコンパイルするコンパイラ

最もシンプルなコンパイラを作ってみよう。1 個の値を入力から読んで、その値を往路グラムの終了コードとして出力するコンパイラを作ってみよう。例えば、入力が 30 の時は以下のようになることが分かる。

```
1  .intel_syntax noprefix
2  .globl main
3
4  main:
5      mov rax, 42
```

```
6       ret
```

1 行目は intel 記法という方法で書くという事を示している。このほかに AT&T 記法という書き方もある。rax というレジスタに値が代入され、ret 命令を実行すると直前に ret に入っていた値をプログラム終了コードとして返すようになっている。

実際に本体を作る。第 1 引数を数値として読み込んで、定型文のアセンブリの中に埋め込むコードは以下のようになる。

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(int argc, char **argv) {
5    if (argc != 2) {
6      fprintf(stderr, "引数の個数が正しくありません\n");
7      return 1;
8    }
9
10   printf(".intel_syntax noprefix\n");
11   printf(".globl main\n");
12   printf("main:\n");
13   printf("  mov rax, %d\n", atoi(argv[1]));
14   printf("  ret\n");
15   return 0;
16 }
```

このプログラムを実行するとアセンブリが生成される。そのアセンブリがうまくいっているかは、アセンブルを使ってアセンブルし、出来上がったものを実行し。”echo $?” と命令を打ち込んでみよう。入力した値が出力されれば、成功である

3.4 まとめ

まず初めに、「低レイヤを知りたい C コンパイラ作成入門」(https://www.sigbus.info/compilerbook#) とその著者である Ueyama Rui 先生に感謝したい。このページは私が今回の記事で書いていたものより細かく、そして発展的な内容もたくさん書いてあるのでこの記事を読んでくれたあなたにはこのページも一読してほしい。はじめての記事投稿、かつテストが迫っている身ではあまりいい記事が書けなかったが、コンパイラ作りについて

私が感じた面白さについてこの記事から少しでも感じてくれたらうれしい。

参考文献

[1] Compiler Explorer https://godbolt.org/
[2] 低レイヤを知りたい C コンパイラ作成入門 https://www.sigbus.info/compilerbook#

編集後記

編集担当の木村です。今回で 61 冊目の Lime になります。

実を言うと、この Lime 61 は 2 年前に集められた記事を Lime 62 を発行するタイミングで編集した部誌です。編集は寺村さんが担当することになっていたのですが、手付かずのまま放置されていたので、私が引き受けました。本当はもう 1 人寄稿してくれた部員がいたのですが、寺村さんの手元には記事が残っておらず、現在は幽霊部員と化しているため、収録は叶いませんでした。Lime 62 では寄稿された記事を全て収録しています。

最後になりましたが、ここまで目を通していただき、ありがとうございました。

令和 5 年 2 月 13 日
編集担当　木村 俊星

Lime Vol. 61

令和 5 年 2 月 13 日発行 第 1 刷

発行 京都工芸繊維大学コンピュータ部
Web サイト: http://www.kitcc.org/
電子メール: question@kitcc.org