DKU Advanced Mobile Lab 1

CA Tutor

Graphic Simulator of MIPS ISA

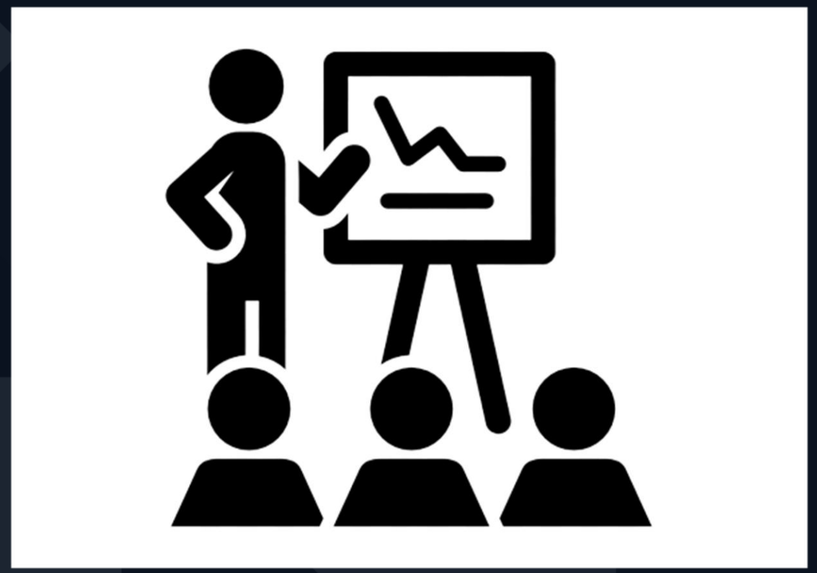
32191097 Kim Junheyong

32192530 Yang Yunseong

32194394 Cho Junghee

32194677 Choi Yunho

Contents



1. Project Introduction
2. Technical Structure
3. Achievements
4. Demonstration

CA Tutor

Graphic Simulator of MIPS ISA

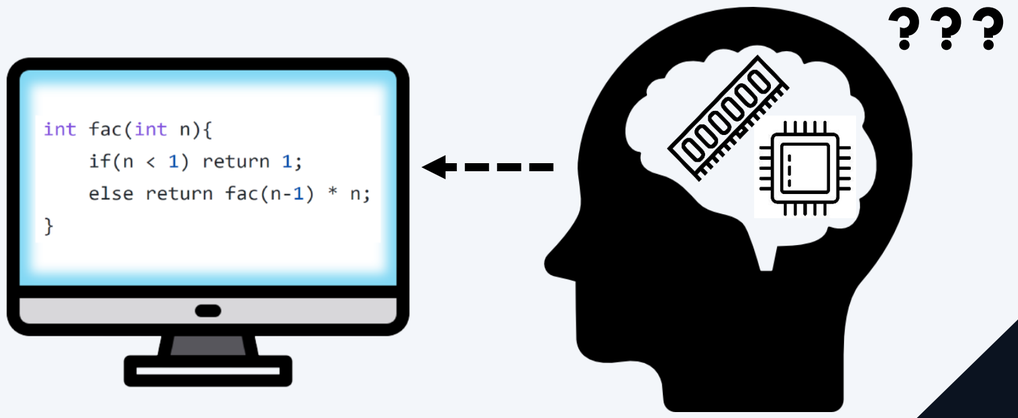
1. Project Introduction

Why we need CA Tutor?

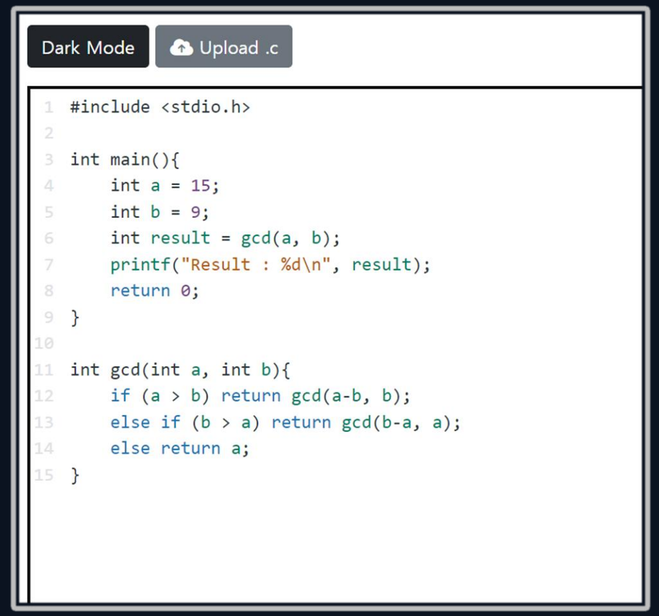
Think about how memory and registers work during program execution. → so difficult.

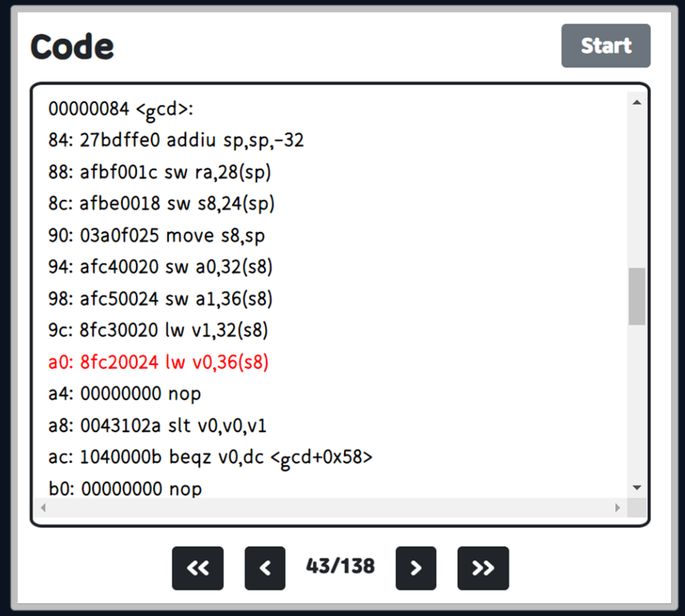
It's hard to learn system programming and computer architecture for the first time.

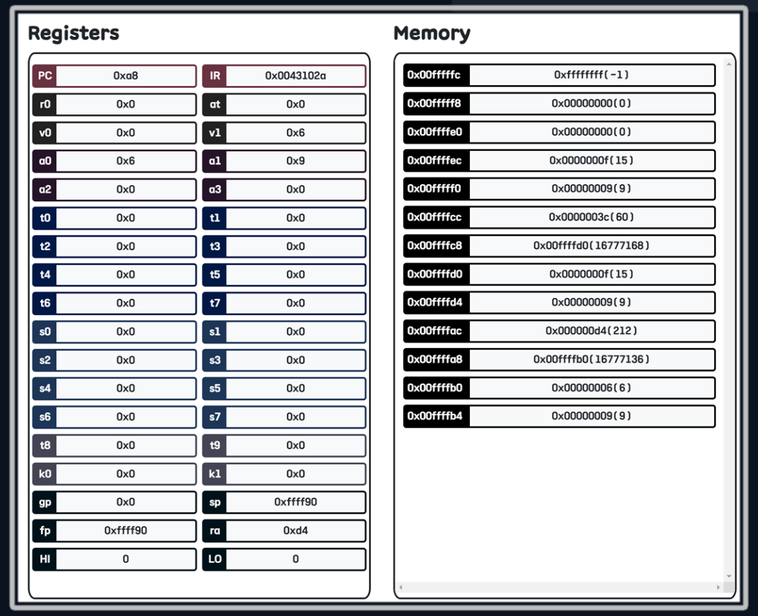
Visualization of memory and register structures is required!



Easy-to-understand MIPS simulator!







User enter C code on our website.
You can write code yourself or import.

CA Tutor changes C code to assembly code and shows it.

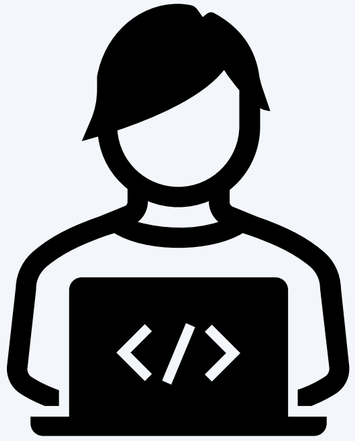
Then check memory and register status step-by-step.

Team Members

MIPS Architecture : Implemented by all members

Choi. Y. H

- UI/UX & PPT
- Front-End



Yang. Y. S

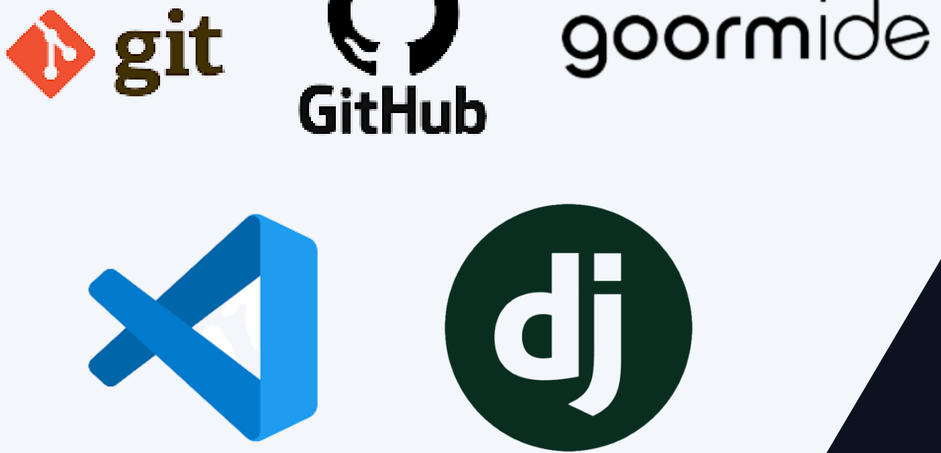
Cho. J. H

- Server & Process
- Event Processing

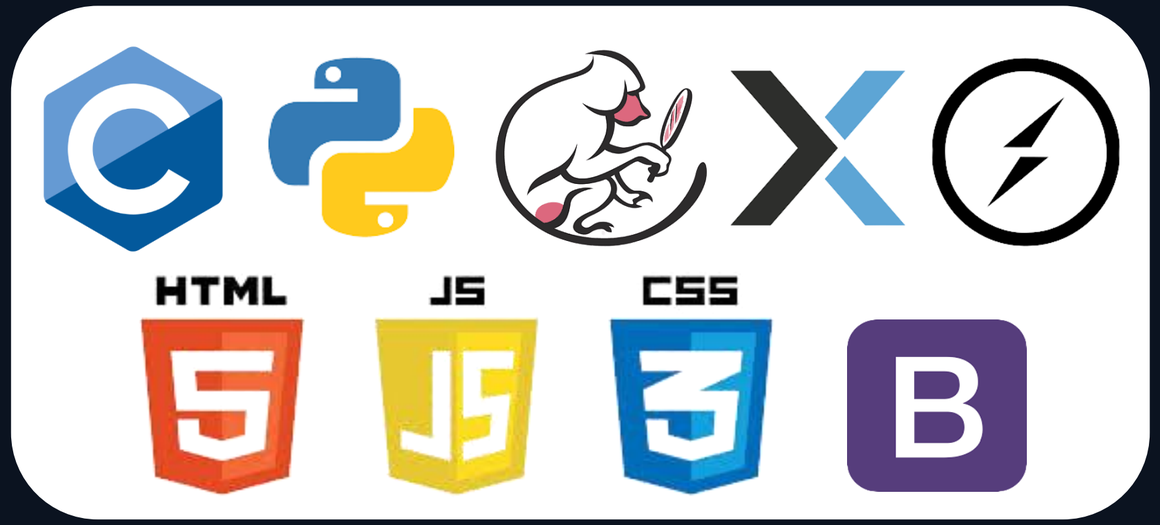
Kim. J. H

2. Technical Structure

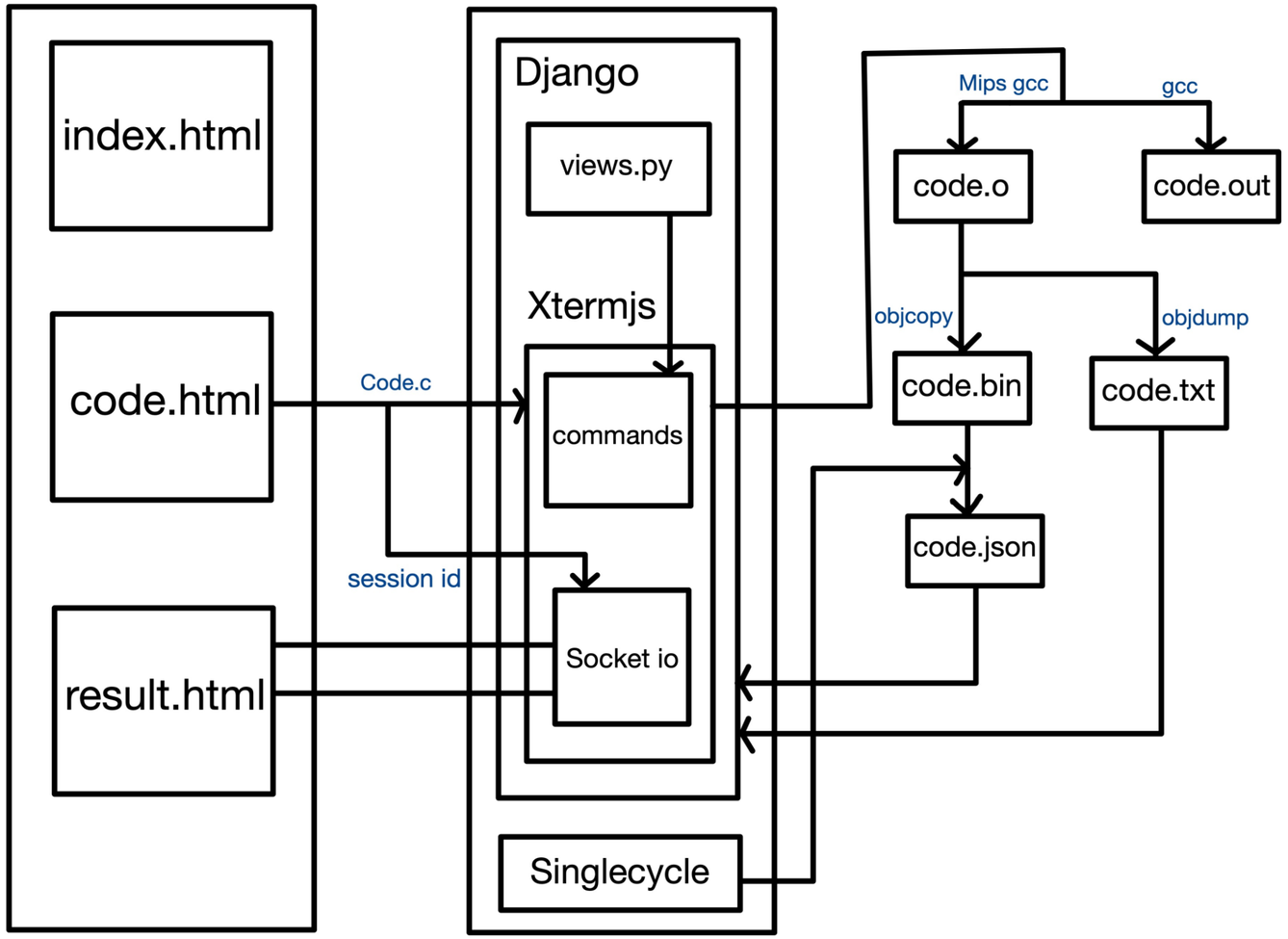
Develop Environment



Tech Stack



System Architecture



Bootstrap : Front-End Framework

1. Responsive Web

Bootstrap helps to easily implementation of responsive web design.

➔ Increase compatibility between different browsers and devices

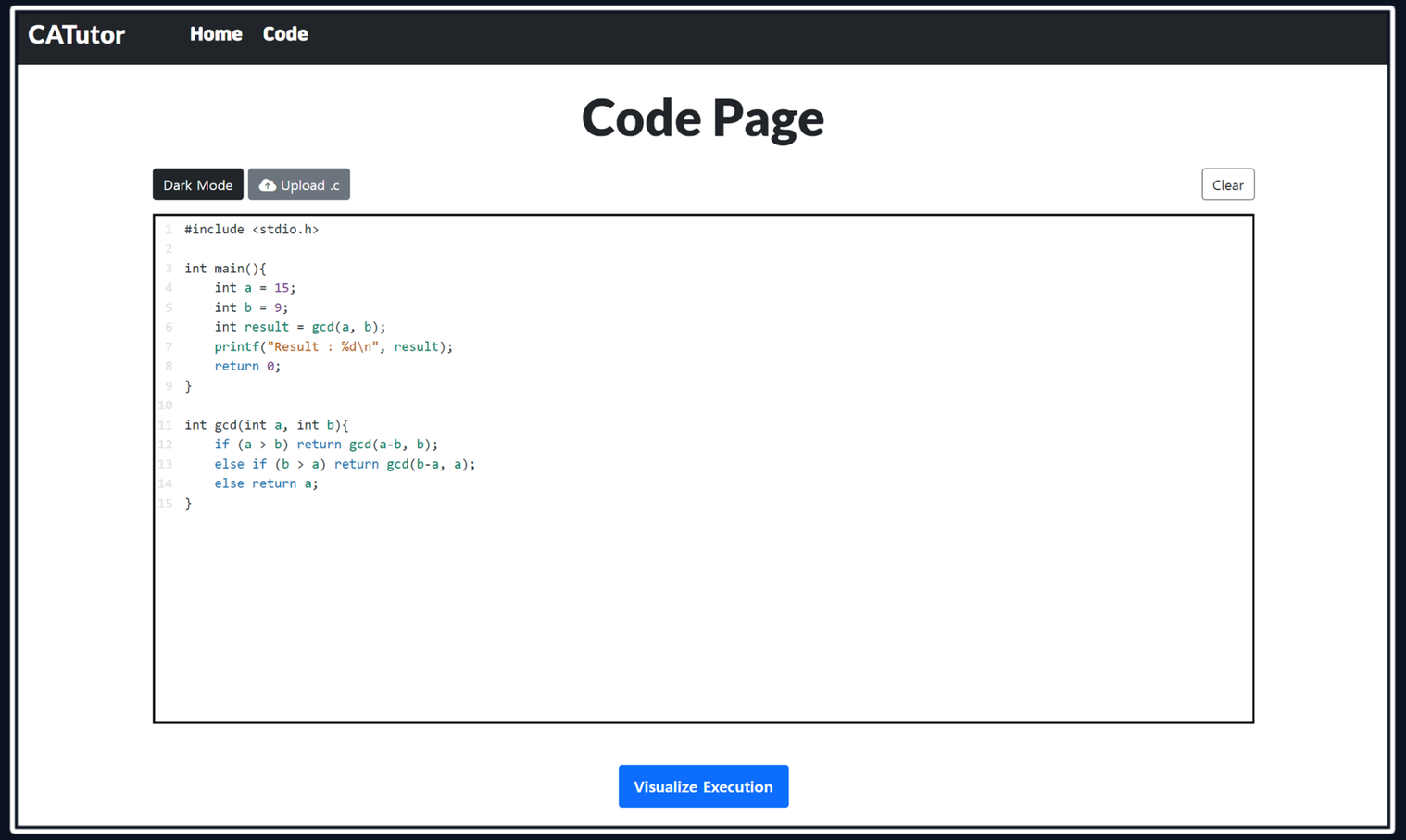
2. UI/UX Design

Bootstrap provides a variety of pre-styled UI components.

➔ Reduce development time and maintain consistent design

Code Page

- Code writing
- Code import
- Send code to server



CodeMirror

- Versatile text editor implemented in JavaScript for use in browsers
- Provides some functions for different programming language
- Provides customizable APIs and plug-in systems → Flexible

Use as C code editor in CA Tutor

```
var editor = CodeMirror.fromTextArea(document.getElementById("codearea"), {
    mode: "text/x-csrc",
    indentWithTabs: true,
    smartIndent: function(state, textBefore) {
        //... 여러 기능 적용
```

- Mode : text/x-scrc (emphasize C language grammar)
- smartIndent : Ability to adjust indentation automatically

Implementation

1. Apply background option (dark mode)

```
function setColorMode() {
    //var textarea = document.getElementById("codearea");
    var button = document.getElementById("color-mode-btn");
    var currentTheme = editor.getOption("theme");
    var newTheme = currentTheme === "neo" ? "3024-night" : "neo";
    editor.setOption("theme", newTheme);
    if (newTheme === "neo") {
        button.innerText = "Dark Mode"
        button.className = "btn btn-dark"
    }
    else {
        button.innerText = "Light Mode"
        button.className = "btn btn-light"
    }
}
```

a) Save dark mode buttons and them objects to variables button and currentTheme

b) Replace with "Neo" ↔ "3024-night" for if-else statement

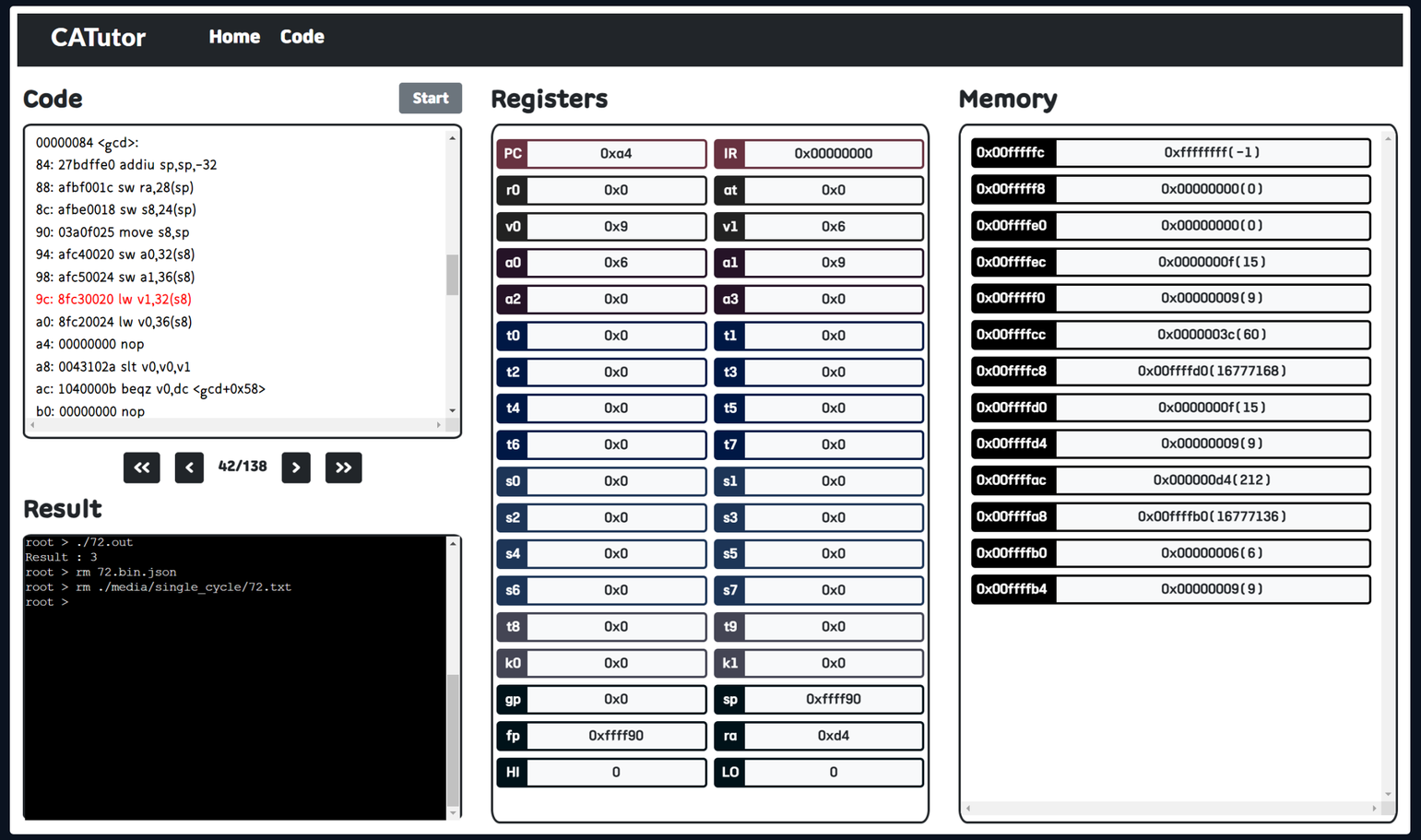
2. Code import (load to code editor)

```
function loadFile() {
    var input = document.getElementById("input-file");
    var file = input.files[0];
    var reader = new FileReader();
    reader.onload = function() {
      editor.setValue(reader.result);
      input.value = "";
    };
    reader.readAsText(file);
}
```

Create a file object imported into FileReader() and write it to the codemirror editor as setValue()

Result Page

- Real-Time Communication
- Apply single-cycle logic
- Reallocate



Xterm & Socket.IO

- Web-based terminal emulator
- Emulates terminal environment in a web browser
- Provides ability for users to enter commands and run programs

- JavaScript library for building real-time web applications
- Enable two-way communication between clients and servers based on web socket protocol.
- Define some events and process them to exchange data

Implementation

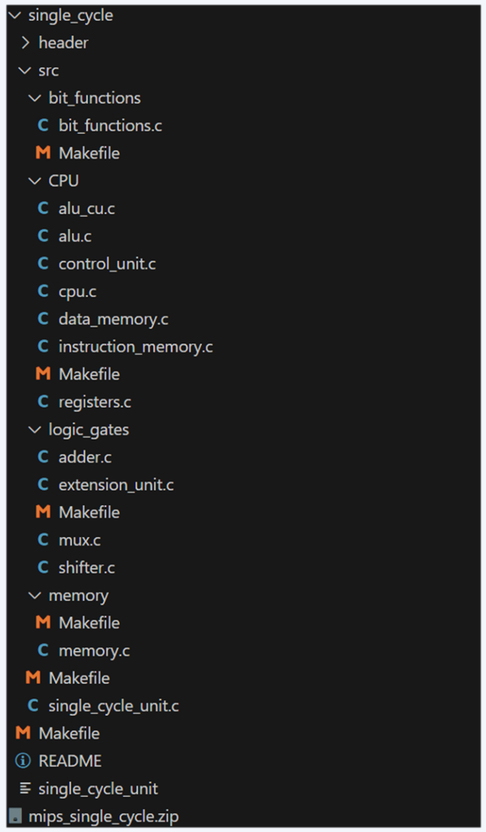
```
Terminal.applyAddon(fit)

var socket = io.connect('https://xtermjs.run.goorm.site',
{transports: ["polling"], extraHeaders: {"test": document.cookie.match('test=([^;]*)')[1]}});

var term = new Terminal({ ···
});

term.open(document.getElementById('terminal'));


socket.on("pty_output", function(output){ ···
})

var dispose = 0;

term.on('key', (key, ev) => { ···
});

socket.on("dispose", function(){ ···
})

var reg_dict = {};


function load_register(){ ···
}

socket.on("send_register", function(output){ ···
})

socket.on("send_code", function(output){ ···
})
```

- Xterm in result.html

```
async_mode = "eventlet"
sio = socketio.Server(async_mode=async_mode,
                      cors_allowed_origins='*')

@sio.event
def resize(sid, message):    ···

@sio.event
def pty_input(sid, message): ···

@sio.event
def load_register(sid): ···

@sio.event
def load_code(sid): ···

@sio.event
def disconnect_request(sid): ···

@sio.event
def connect(sid, environ): ···

@sio.event
def disconnect(sid): ···
```

- Socket.IO in views.py

Convert MIPS and Compile



- Single-Cycle Directory

```
os.write(fd, "export PS1=\"\\u > \"\n".encode())
os.write(fd, "mips-linux-gnu-gcc -c media/{}.c -mips1 -mfp32\n".format(test).encode())
os.write(fd, "mips-linux-gnu-objcopy -O binary -j .text {}.o {}.bin\n".format(test, fd).encode())
os.write(fd, "readelf -r {}.o > {}_rel.txt\n".format(test, fd).encode())
os.write(fd, "python test.py {}\n".format(fd).encode())
os.write(fd, "cd media/single_cycle\n".encode())
os.write(fd, "chmod +x single_cycle_unit\n".encode())
os.write(fd, "./single_cycle_unit ../../{}.bin\n".format(fd).encode())
os.write(fd, "mips-linux-gnu-objdump -dS ../../{}.o > {}.txt\n".format(test, fd).encode())
os.write(fd, "rm ../../{}.o\n".format(test).encode())
os.write(fd, "rm ../../{}.bin\n".format(fd).encode())
os.write(fd, "rm ../../{}_rel.txt\n".format(fd).encode())
os.write(fd, "cd ../../\n".encode())
os.write(fd, "gcc media/{}.c -o {}.out\n".format(test, fd).encode())
os.write(fd, "rm media/{}.c\n".format(test).encode())
os.write(fd, "clear\n".encode())
os.write(fd, "./{}.out\n".format(fd).encode())
```

- Event process function connect()

  Compile to os.write on Xterm terminal (variable fd)

  MIPS cross compile, gcc, etc.

Print Single-Cycle Data

```c
sprintf(s1, "%s.json", argv[1]);

freopen(s1, "w", stdout);
printf("{\"total\": [{\"cycle\": 0, \"pc\": \"0x%x\", \"ir\": \"0x00000000\"", pc);
for (int i = 0; i<32; i++){
    printf(", \"reg[%d]\": \"0x%x\"", i, reg[i]);
}
printf("}\n");

while (pc != 0xffffffff){
    fetch(&cpu, mem);
    decode(&cpu);
    execute(&cpu);
    memory_operation(&cpu, mem);
    write_back(&cpu, mem);
    pc_update(&cpu);
    print_result();
}
printf("]}");
//print_result();
fclose(stdout);
```

- single_cycle_unit.c

  Write each cycle's register and memory status in .json file

```python
@sio.event
def load_register(sid):
    global dic
    fd = dic.get(sid)[1]
    data={}

    file_name = str(dic.get(sid)[1])+".bin.json"


    with open(file_name, 'r') as file:
        data = json.load(file)
        data["sid"] = sid

    sio.emit("send_register", data)
    os.write(fd, "rm {}.bin.json\n".format(fd).encode())
```

- Event process function load_register()

  Send .json file as dictionary form

Reallocate

```
file_name = sys.argv[1]
txtfile = file_name + "_rel.txt"
binfile = file_name + ".bin"

f = open(txtfile,"r")
lines = f.readlines()
f.close()

isText = False
relText = []
for i in lines:
    if i == "\n":
        isText = False
    if isText:
        if i.find("Offset") != -1:
            continue
        relText.append(i)
    if i.find(".rel.text" ) != -1:
        isText = True

arr = []

for i in relText:
    tmp = i.split()
    arr.append( ((int(tmp[0],16)),(int(tmp[3],16))) )
```

- Process _rel.txt header information which contain reallocation data

```
f = open(binfile, "rb")
a = bytearray(f.read())

f.close()

f = open(binfile, "wb")
for i in arr:

    index = i[0]
    adr = i[1] >>2
    ins = (a[index]<<24) + (a[index+1]<<16) + (a[index+2]<<8) + (a[index+3])
    opcode = ins & 0xfc000000
    tmp = opcode >> 26
    if (tmp == 0x2) or (tmp == 0x3):
        relins = opcode + adr
    else:
        relins = (ins & 0xffff0000) + i[1]


    a[index] = (relins >> 24) & 0xff
    a[index+1] = (relins >> 16) & 0xff
    a[index+2] = (relins >> 8) & 0xff
    a[index+3] = (relins) & 0xff

f.write(a)
f.close()
```

- Modify bin file with processed information

3. Achievements

Initial goal

- ✓ Implement single-cycle structure
- ✓ Save code and convert MIPS
- ✓ Real-time communication
- ✓ Print result step-by-step
- ✓ Create graphic simulator
- ✓ Complete to make web page
- ▲ Login and sign-up

Accomplishment %

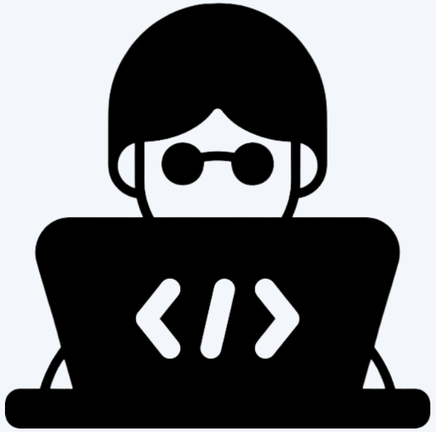
Team technical achievements

1. Study about computer architecture
   (Especially MIPS ISA and Single-Cycle)

2. Improve a variety of language skills
   (C, Python, HTML, CSS, JavaScript)

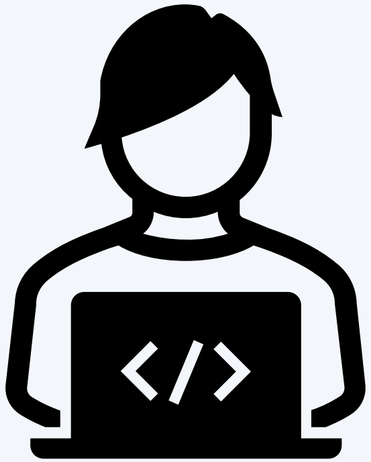
3. Learn many new technologies
   (Django, Bootstrap, Xterm, SocketIO · · ·)

Personal Lessons



When doing a project, I realized that it is necessary to respect each other's ideas and opinions, and through this, we can make the most of each other's strengths.

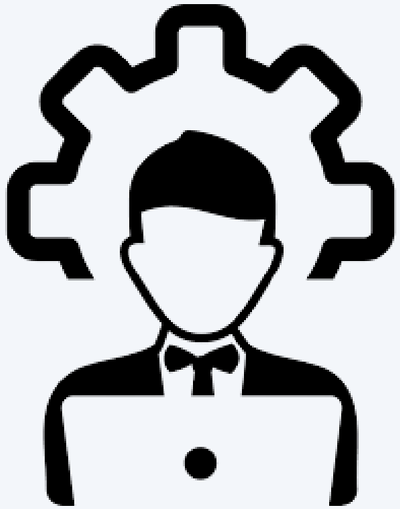
Choi. Y. H



I was able to improve my front-end development skills and had deep understanding of the computer's instruction execution method.

Yang. Y. S

Personal Lessons



I realized the importance of clearly expressing my thoughts and handling exceptions. And I could clearly learn the structure and use of the elf file.

Cho. J. H

I was able to gain various knowledge by working with my team members on troubles shooting, and now I think I will be able to design project more specifically.

Kim. J. H

4. Demonstration

Github Repository

(https://github.com/judemario-kim/ca-tutor)

Our Website

(https://xtermjs.run.goorm.site/catutor/)

It works well on smart phone, too.

DKU Advanced Mobile Lab 1

Thank You

For Your Attention

CA Tutor

Graphic Simulator of MIPS ISA