

컴퓨터 조직론

Project 2

Jaewon Hur

System & software security Lab.

Seoul National University

Mar. 16, 2020

목차

- 프로젝트 목표
- 프로젝트 개요
- 프로젝트 구성
- 평가 기준
- 참고

프로젝트 목표

■ 전체 목표

- Chisel 언어를 이용해 간단한 In-order CPU를 구현.
- 앞으로 총 3개의 프로젝트를 통해 최종적으로 multi-cycle CPU를 구현하게 됨.
 1. Single-cycle CPU R-type instructions 구현
 2. Single-cycle CPU 전체 구현
 3. Multi-cycle CPU 구현

■ 본 프로젝트 목표

- Single-cycle CPU R-type instruction 구현
 1. ALU control 구현
 2. R-type instruction을 위한 diagram 작성
 3. ADD instruction 구현
 4. 나머지 R-type instruction 구현

프로젝트 개요

■ R-type^[1]

- Integer Register-Register operation
- RV32I 에서 정의하는 가장 기본적인 instruction type
- ADD/SLT/SLTU/AND/OR/XOR/SLL/SRL/SUB/SRA operation
- Format

31	25 24	20 19	15 14	12 11	7 6	0
funct7	rs2	rs1	funct3	rd	opcode	
7	5	5	3	5	7	
0000000	src2	src1	ADD/SLT/SLTU	dest	OP	
0000000	src2	src1	AND/OR/XOR	dest	OP	
0000000	src2	src1	SLL/SRL	dest	OP	
0100000	src2	src1	SUB/SRA	dest	OP	

Fig1. R-type format

[1] Waterman, Andrew; Asanović, Krste; SiFive Inc. "The RISC-V Instruction Set Manual, Volume I: Unprivileged ISA". University of California, Berkeley. Retrieved 13 Dec 2019, p.g. 19

프로젝트 개요

■ R-type^[1]

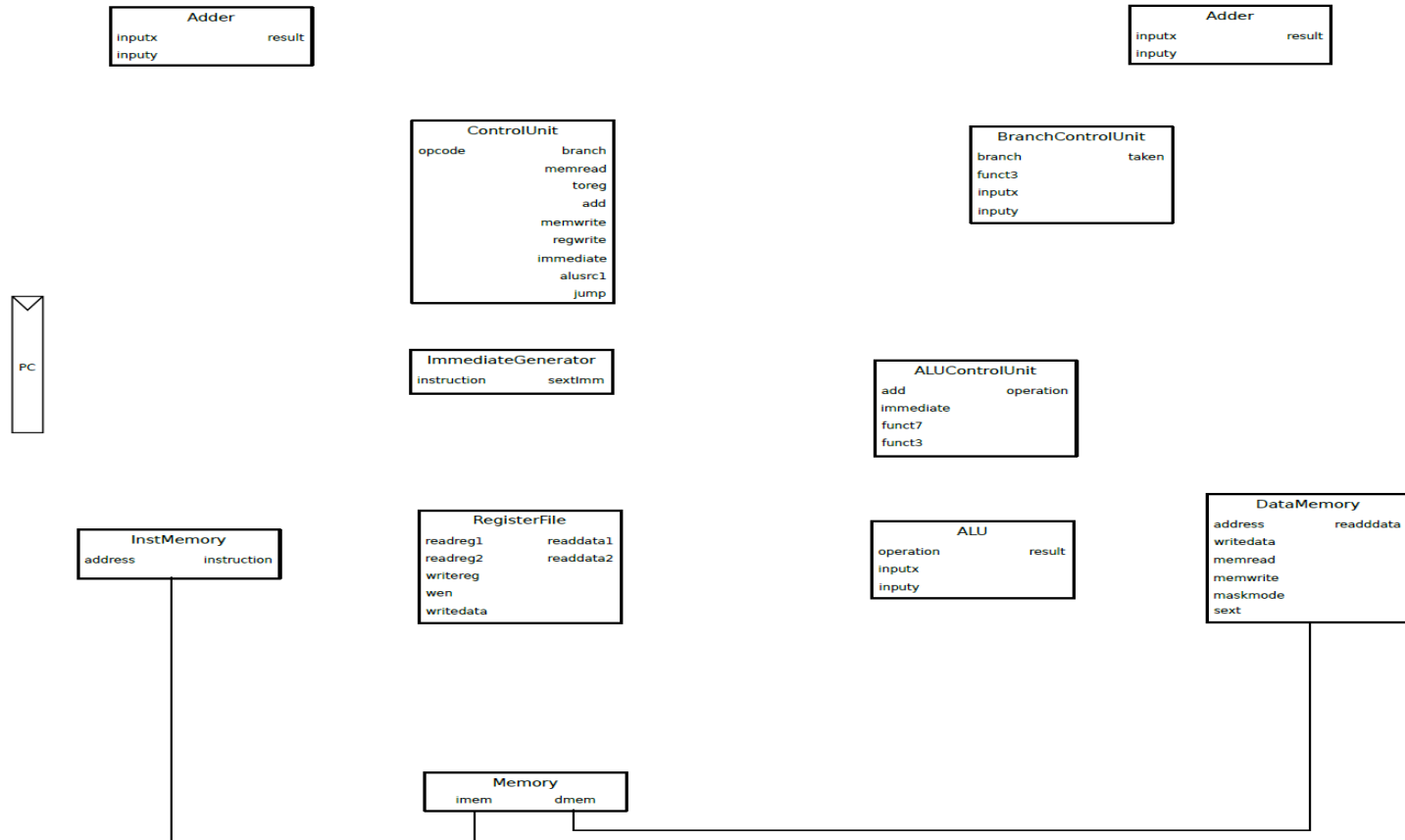


Fig1. Skeleton diagram

[1] Waterman, Andrew; Asanović, Krste; SiFive Inc. "The RISC-V Instruction Set Manual, Volume I: Unprivileged ISA". University of California, Berkeley. Retrieved 13 Dec 2019, p.g. 19

프로젝트 구성

2. ALU control 구현

- 주어진 code에 ALU는 이미 구현되어 있음.
- Instruction에 따라 ALU에 알맞은 입력 (*operation*) 을 출력하는 ALU control을 구현.
 - src/main/scala/components/alucontrol.scala** 수정 (skeleton code는 제공됨)
- Inputs to ALU: *operation*, *inputx*, *inputy* (3 inputs)
 - inputx*, *inputy*는 이미 연결되어 있음
- Inputs to ALU control: *add*, *immediate*, *funct7*, *funct3* (4 inputs)
 - add*, *immediate*는 control unit로부터 입력, *funct7*, *funct3*는 instruction의 일부
 - Instruction format은 RISC-V instruction set manual^[1] 130p.g.에서 찾을 수 있음.
 - 지금은 *add*, *immediate* field 무시 (둘다 false로 가정).
- 평가 및 검증
 - sbt shell 안에서 다음 명령어 입력

sbt> Lab1 / testOnly dinocpu.ALUControlTesterLab1

- 실행 결과

```
[info] ALUControlTesterLab1:
[info] ALUControl
[info] - should match expectations for each instruction type
[info] ScalaTest
[info] Run completed in 763 milliseconds.
[info] Total number of tests run: 1
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 1, failed 0, canceled 0, ignored 0, pending 0
[info] All tests passed.
[info] Passed: Total 1, Failed 0, Errors 0, Passed 1
[success] Total time: 1 s, completed Feb 9, 2020 8:25:14 AM
```

0000	and
0001	or
0010	add
0011	sub
0100	slt
0101	sltu
0110	sll
0111	srl
1000	sra
1001	xor

Fig2. Operation field

프로젝트 구성

1. R-type instruction을 위한 diagram 작성

- ALU와 ALU control을 구현하였으므로, 나머지 part에서 각 unit들을 올바른 wire로 연결함.
- 우선 CPU 내부 구성에 대한 큰 그림 (diagram)을 작성.
- 주의 사항
 - 주어진 모든 module을 연결할 필요 없음. (R-type과 관련된 wire만 연결)
 - ALUControlUnit의 add, immediate port는 연결할 필요 없음.
 - Wire마다 bit width를 표기.
 - Wire가 signal의 일부분일 경우 어느 부분인지 명시
- 평가 및 검증
 - Wire 연결이 올바른가?
 - Bit width가 정확한가?

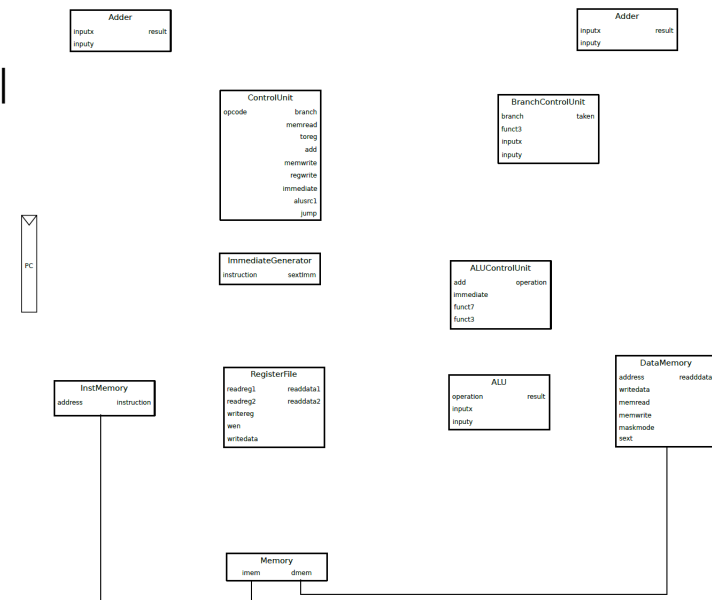


Fig3. Skeleton diagram

프로젝트 구성

3. ADD instruction 구현

- 앞서 작성한 diagram에 따라 각 module들 연결
 - src/main/scala/single-cycle/cpu.scala**에 *Memory, InstMemory, DataMemory*를 제외한 module들이 *Chisel 'Module'*로 정의되어 있음
 - ADD instruction에 필요한 wire들을 연결
 - 예시)
 - `io.imem.address := pc` 는 Fig. 4의 wire연결을 나타냄
- 주의 사항
 - Module 연결이 완료되면 DontCare 구문을 지움
 - `//debug / pipeline viewer` 밑으로는 수정 금지
- 평가 및 검증
 - sbt shell 안에서 다음 명령어 입력

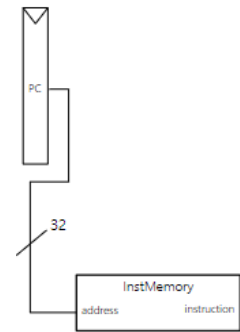


Fig4. pc wire

sbt> Lab1 / testOnly dinocpu.SingleCycleAddTesterLab1

- 실행 결과

```
CYCLE=1
pc: 4
control: AnonymousBundle(opcode -> 0, branch -> 0, memread -> 0, toreg -> 3, add -> 0, memwrite -> 0, regwrite -> 0, immediate -> 0, alusrc1 -> 0, jump -> 0)
registers: AnonymousBundle(readreg1 -> 0, readreg2 -> 0, writereg -> 0, writedata -> 0, wen -> 0, readdatal -> 0, readdatal2 -> 0)
aluControl: AnonymousBundle(add -> 0, immediate -> 0, funct7 -> 0, funct3 -> 0, operation -> 2)
alu: AnonymousBundle(operation -> 2, inputx -> 0, inputy -> 0, result -> 0)
immGen: AnonymousBundle(instruction -> 0, sextImm -> 0)
branchCtrl: AnonymousBundle(branch -> 0, funct3 -> 0, inputx -> 0, inputy -> 0, taken -> 0)
pcPlusFour: AnonymousBundle(inputx -> 4, inputy -> 4, result -> 8)
branchAdd: AnonymousBundle(inputx -> 0, inputy -> 0, result -> 0)

[info] SingleCycleAddTesterLab1:
[info] Single Cycle CPU
[info] - should run add test add1
[info] ScalaTest
[info] Run completed in 2 seconds, 145 milliseconds.
[info] Total number of tests run: 1
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 1, failed 0, canceled 0, ignored 0, pending 0
[info] All tests passed.
[info] Passed: Total 1, Failed 0, Errors 0, Passed 1
[success] Total time: 4 s, completed Feb 9, 2020 8:23:30 AM
```


프로젝트 구성

3. 나머지 R-type instruction 구현

- 앞서 구현한 ALU control과 ADD instruction의 test가 성공하면 나머지 R-type instruction도 동작함.
 - 실패할 경우, ALU control 혹은 module 연결에 문제.
- 고려 사항
 - Add0 test는 reg 0에 값을 적는지 검증 (RISC-V isa에서 reg0는 항상 0으로 고정)
- 평가 및 검증

sbt> Lab1 / testOnly dinocpu.SingleCycleRTypeTesterLab1

sbt> Lab1 / testOnly dinocpu.SingleCycleMultiCycleTesterLab1

```
[info] SingleCycleRTypeTesterLab1:
[info] Single Cycle CPU
[info] - should run R-type instruction add1
[info] - should run R-type instruction add2
[info] - should run R-type instruction add0
[info] - should run R-type instruction or
[info] - should run R-type instruction sub
[info] - should run R-type instruction and
[info] - should run R-type instruction xor
[info] - should run R-type instruction slt
[info] - should run R-type instruction slti
[info] - should run R-type instruction sltu
[info] - should run R-type instruction sltui
[info] - should run R-type instruction sll
[info] - should run R-type instruction srl
[info] - should run R-type instruction sra
[info] ScalaTest
[info] Run completed in 8 seconds, 167 milliseconds.
[info] Total number of tests run: 14
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 14, failed 0, canceled 0, ignored 0, pending 0
[info] All tests passed.
[info] Passed: Total 14, Failed 0, Errors 0, Passed 14
[success] Total time: 9 s, completed Feb 9, 2020 8:26:22 AM
```

```
[info] SingleCycleMultiCycleTesterLab1:
[info] Single Cycle CPU
[info] - should run R-type multi-cycle program addfwd
[info] - should run R-type multi-cycle program swapxor
[info] - should run R-type multi-cycle program power2-512
[info] - should run R-type multi-cycle program power2-1234
[info] - should run R-type multi-cycle program power2--65536
[info] - should run R-type multi-cycle program oppsign-true
[info] - should run R-type multi-cycle program oppsign-false
[info] - should run R-type multi-cycle program rotR
[info] ScalaTest
[info] Run completed in 5 seconds, 198 milliseconds.
[info] Total number of tests run: 8
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 8, failed 0, canceled 0, ignored 0, pending 0
[info] All tests passed.
[info] Passed: Total 8, Failed 0, Errors 0, Passed 8
[success] Total time: 6 s, completed Feb 9, 2020 8:28:13 AM
```

평가 기준

1. ALU control 구현

- 주어진 test case 성공 여부에 따라 100점 혹은 0점

2. R-type instruction을 위한 diagram 작성

- Wire 연결과 signal 명시, bit width가 모두 올바른 경우: 100점
- Bit width에 오류가 있는 경우: 50점
- Wire 연결 혹은 signal에 오류가 있는 경우: 0점

3. ADD instruction 구현

- 주어진 test case 성공 여부에 따라 만점 혹은 0점

4. 나머지 R-type instruction 구현

- 성공하는 test case 개수에 따라 차등

참고

1. Printf debugging

- Simulation 도중 변수를 확인하고 싶을 때, printf를 사용한다.
 - 매 cycle printf가 출력된다.
 - 예제)
 - `printf(p“This is my text with a $var\n”), printf(p“Output: ${io.output}”)`
 - 따옴표 앞 p를 빼먹지 않도록 주의.
- Compilation 중 변수를 확인하고 싶을 때, println을 사용한다.
 - 예제)
 - `println(s“This is my variable: $var\n”)`

프로젝트 진행 과정

1. proj2_hw.tar 압축파일 다운로드 (etl 홈페이지에서 다운)
2. proj2_hw.tar 파일을 dinocpu.box와 같은 폴더에 이동
3. 'tar xvf proj2_hw.tar' 명령으로 압축해제 (proj2_hw 폴더가 생김)
4. 'vagrant up' & 'vagrant ssh'로 dinocpu vagrant 접속
5. vagrant 안에서 'cd dinocpu/proj2_hw'로 '~/dinocpu/proj2_hw' 폴더로 이동
6. 'singularity run library://jlowepower/default/dinocpu' 를 수행하여 singularity 이미지 파일 다운로드 및 singularity 환경으로 접속 (>sbt shell로 들어가게 됨)
7. 이 다음은 앞서 설명한대로 source code를 채우고, 프로젝트 진행

프로젝트 진행 과정 1 & 2

tmux

```
~/Class/Assistant/Lab2
ls
dinocpu.box  proj2_hw.tar  vagrant  Vagrantfile

~/Class/Assistant/Lab2
_
```

tar xvf proj2_hw.tar

tmux

```
~/Class/Assistant/Lab2
ls
dinocpu.box  proj2_hw  proj2_hw.tar  vagrant  Vagrantfile

~/Class/Assistant/Lab2
_
```

프로젝트 진행 과정 3

vagrant up; vagrant ssh

```
tmux
~/Class/Assistant/Lab2
vagrant up; vagrant ssh
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Clearing any previously set forwarded ports...
==> default: Fixed port collision for 22 => 2222. Now on port 2200.
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 22 (guest) => 2200 (host) (adapter 1)
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2200
    default: SSH username: vagrant
    default: SSH auth method: private key
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
    default: The guest additions on this VM do not match the installed version of
    default: VirtualBox! In most cases this is fine, but in rare cases it can
    default: prevent things such as shared folders from working properly. If you see
    default: shared folder errors, please make sure the guest additions within the
    default: virtual machine match the version of VirtualBox you have installed on
    default: your host and reload your VM.
    default:
    default: Guest Additions Version: 6.0.4
    default: VirtualBox Version: 5.2
==> default: Mounting shared folders...
    default: /vagrant => /home/jwhur/Class/Assistant/Lab2
==> default: Machine already provisioned. Run `vagrant provision` or use the `--provision`
==> default: flag to force provisioning. Provisioners marked to run always will still run.
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-45-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Last login: Sat Apr 18 13:19:49 2020 from 10.0.2.2
ln: failed to create symbolic link '/home/vagrant/dinocpu/vagrant': File exists
vagrant@vagrant:~$ ls
dinocpu
vagrant@vagrant:~$
```

프로젝트 진행 과정 4 & 5

cd dinocpu/proj2_hw

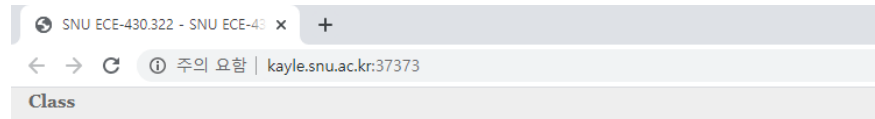
```
tmux
vagrant@vagrant:~$ cd dinocpu/proj2_hw/
vagrant@vagrant:~/dinocpu/proj2_hw$ ls
build.sbt  LICENSE  project  src  Vagrantfile
vagrant@vagrant:~/dinocpu/proj2_hw$
```

singularity run library://jlowepower/default/dinocpu

```
tmux
vagrant@vagrant:~/dinocpu/proj2_hw$ singularity run library://jlowepower/default/dinocpu
WARNING: Authentication token file not found : Only pulls of public images will succeed
[info] Loading settings for project proj2_hw-build from plugins.sbt ...
[info] Loading project definition from /vagrant/proj2_hw/project
[info] Loading settings for project root from build.sbt ...
[info] Set current project to dinocpu (in build file:/vagrant/proj2_hw/)
[info] sbt server started at local:///home/vagrant/.sbt/1.0/server/e1c97ddd91a08057f6e1/sock
sbt:dinocpu>
```

제출 방법

1. 제출 홈페이지 접속 (<http://kayle.snu.ac.kr:37373/>)
2. 자신의 snu 이메일 등록 (snu 계정 외 이메일은 사용 불가), 이메일 등록시 해당 메일로 api-key가 전송됨.



SNU ECE-430.322

Register

hurjaewon@snu.ac.kr

Email api-key

- Register your email (Only @snu.ac.kr emails are allowed).
- Email will be sent with an api-key.

Login

your_api_key

Submit api-key

- Enter api-key to login

Contact: byoungyoung@snu.ac.kr

제출 방법

3. 이메일로 받은 api-key로 홈페이지 접속

SNU ECE-430.322 - SNU ECE-430.322 x +

← → ↻ ⚠ 주의 요함 | kayle.snu.ac.kr:37373

Class

SNU ECE-430.322

Register

- Register your email (Only @snu.ac.kr emails are allowed).
- Email will be sent with an api-key.

Login

- Enter api-key to login

Contact: byoungyoung@snu.ac.kr

Submission - SNU ECE-430.322 x +

← → ↻ ⚠ 주의 요함 | kayle.snu.ac.kr:37373/student

Class | New api-key | Logout

SNU ECE-430.322

Submission for hurjaewon@snu.ac.kr

Submission

선택된 파일 없음

- Your api-key (hurjaewon@snu.ac.kr): **gDWUMYUPTeJU7JZ7KRWWKCFXNoNVPE4D**
- When uploading, you should strictly follow the file naming rules. Otherwise, the server will not accept the submission.
Lab: **lab[o-g] + .zip or lab[o-g] + .tar.gz**
e.g., to upload your lab1, the filename should be either **lab1.zip** or **lab1.tar.gz**
- If you want to upload through the terminal (e.g., lab1.zip), you can do the following.

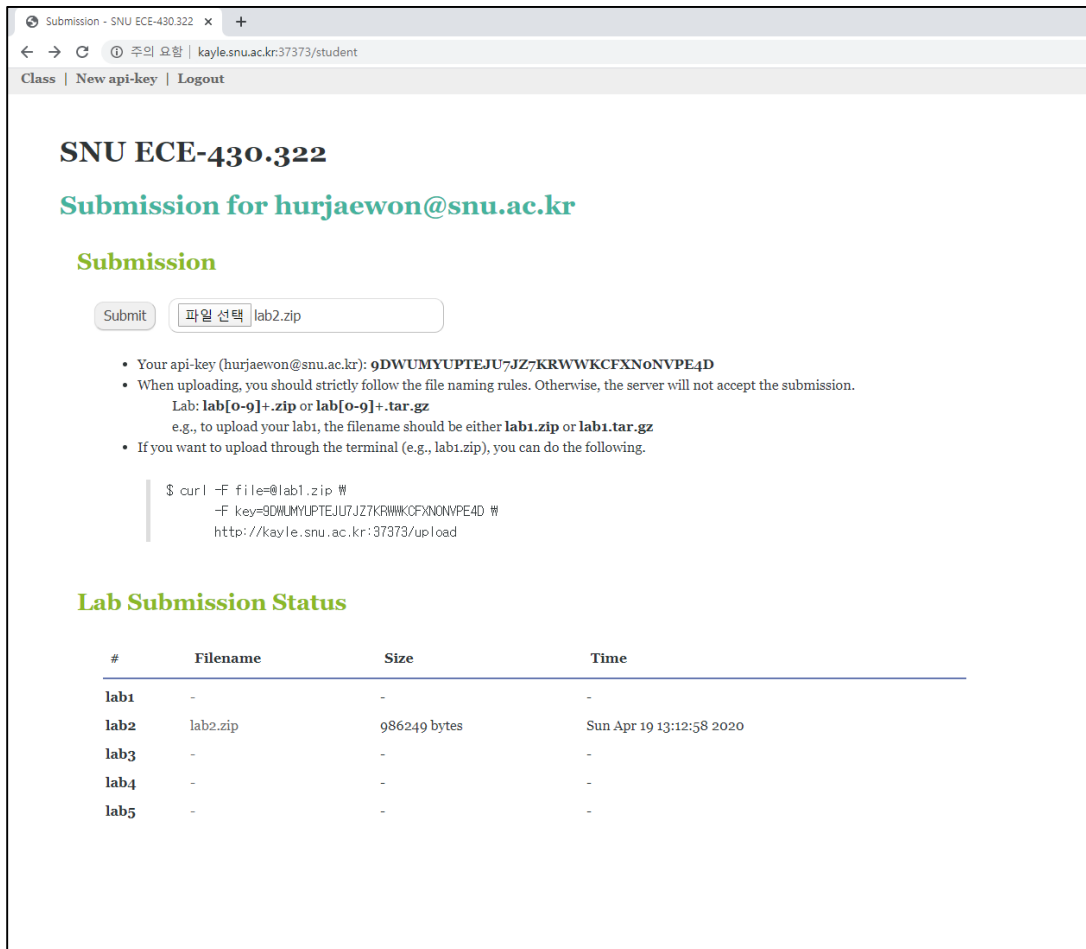
```
$ curl -F file=@lab1.zip \
-F key=gDWUMYUPTeJU7JZ7KRWWKCFXNoNVPE4D \
http://kayle.snu.ac.kr:37373/upload
```

Lab Submission Status

#	Filename	Size	Time
lab1	-	-	-
lab2	-	-	-
lab3	-	-	-
lab4	-	-	-
lab5	-	-	-

제출 방법

3. proj2_hw 폴더에서 'zip lab2.zip proj2_cpu_diagram.pdf src' 명령으로 diagram파일과 src 폴더를 lab2.zip 파일로 압축한 후 홈페이지에 제출



Submission - SNU ECE-430.322

← → ↻ 주의 요함 | kayle.snu.ac.kr:37373/student

Class | New api-key | Logout

SNU ECE-430.322

Submission for hurjaewon@snu.ac.kr

Submission

Submit

- Your api-key (hurjaewon@snu.ac.kr): **9DWUMYUPTJ7JZ7KRWWKCFXNoNVPE4D**
- When uploading, you should strictly follow the file naming rules. Otherwise, the server will not accept the submission.
Lab: **lab[o-9]+.zip** or **lab[o-9]+.tar.gz**
e.g., to upload your lab1, the filename should be either **lab1.zip** or **lab1.tar.gz**
- If you want to upload through the terminal (e.g., lab1.zip), you can do the following.

```
$ curl -F file=@lab1.zip \
-F key=9DWUMYUPTJ7JZ7KRWWKCFXNoNVPE4D \
http://kayle.snu.ac.kr:37373/upload
```

Lab Submission Status

#	Filename	Size	Time
lab1	-	-	-
lab2	lab2.zip	986249 bytes	Sun Apr 19 13:12:58 2020
lab3	-	-	-
lab4	-	-	-
lab5	-	-	-

제출기한은 5/11 (월요일) 오후 11:55까지 입니다.
(delay 제출 패널티는 추후 공지하겠습니다.)

Thank you

문의사항은 hurjaewon@snu.ac.kr 로 혹은 etl 게시판에 문의 바랍니다.