MIPS Verification

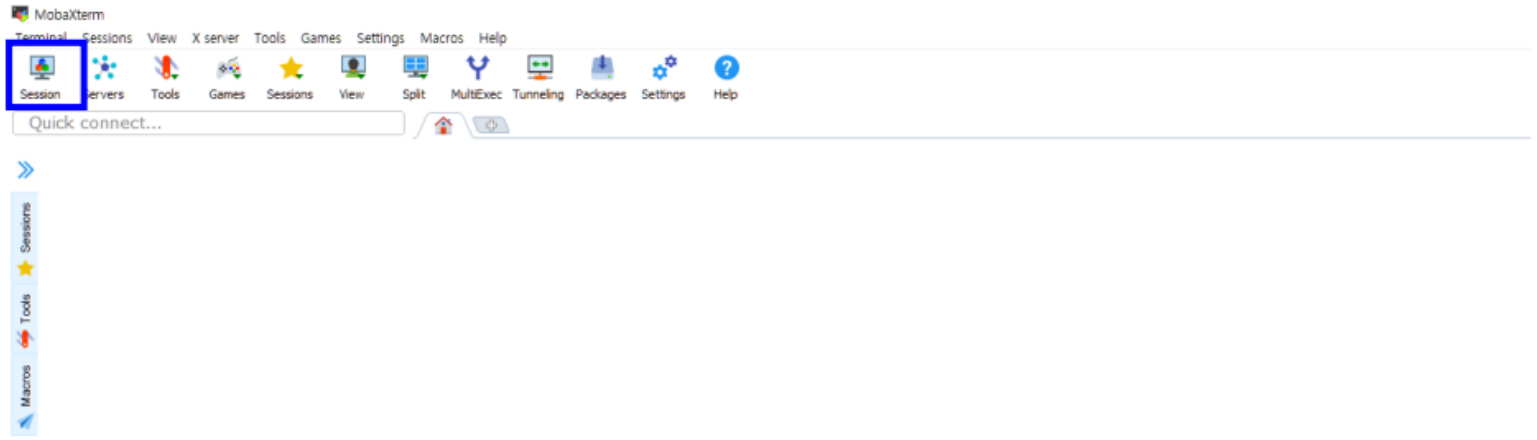
조교 박현재

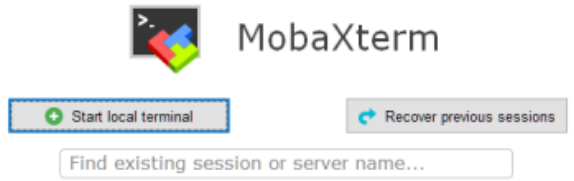
개요

- 서버 안내
- iVerilog를 통한 Testbench 빌드
- Verification with GTKWave
- Makefile 생성

서버 안내

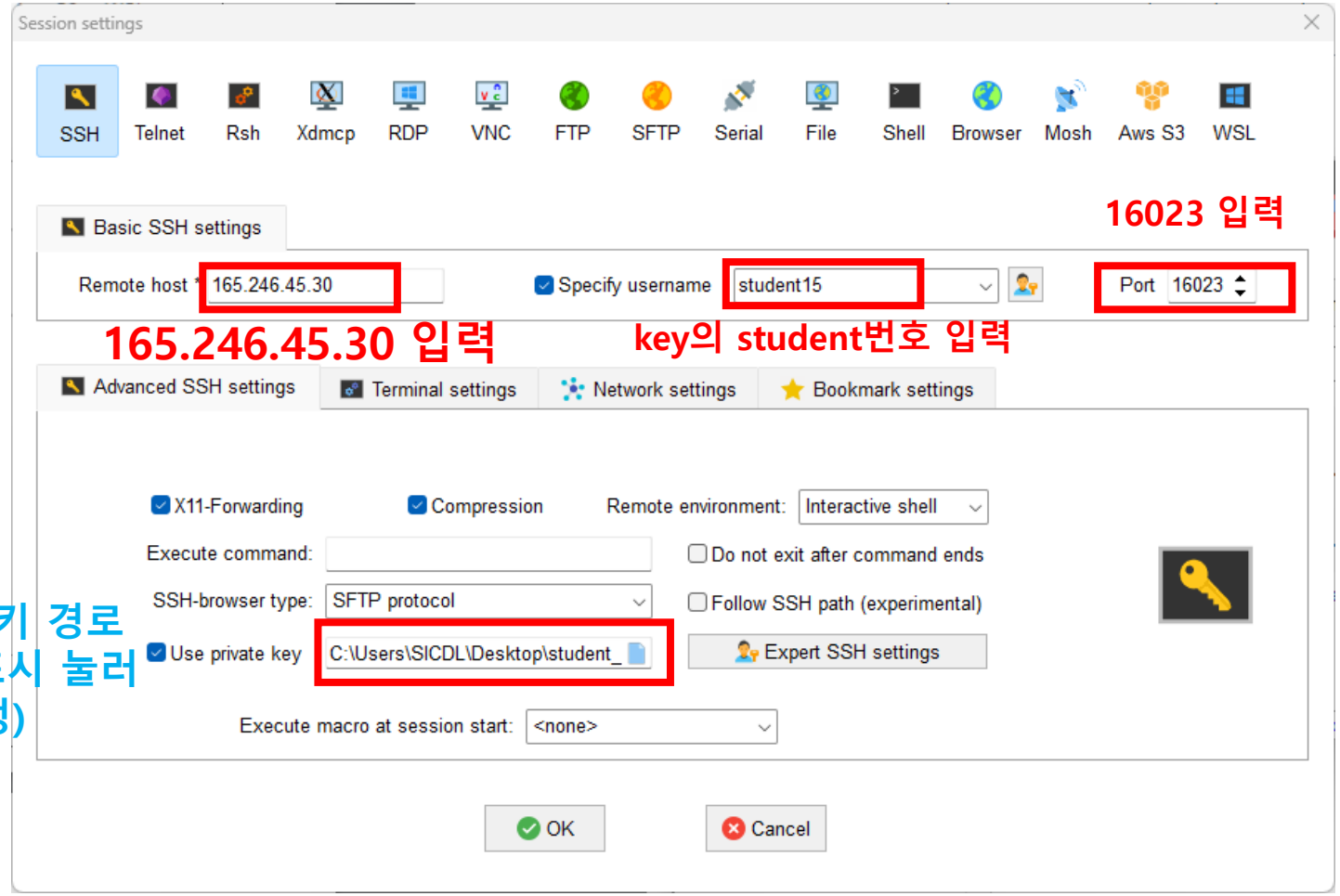
- SSH 접속
  - Mobaxterm 다운로드
    https://download.mobatek.net/2232022120824733/MobaXterm_Installer_v22.3.zip
  - Mobaxterm 설치 후 실행 → Session 클릭





서버 안내

- SSH 접속



제공받은 개인키 경로
(파란색 문서 표시 눌러
경로 지정)

Test 할 Assembly

```
# Test whether operations which require a stall and
# forward work correctly.
#

# initial values
addi $t0, $zero, 5
addi $t1, $zero, 7
addi $t2, $zero, 9

# store words in memory
sw $t0, 0($zero)  # 5
sw $t1, 4($zero)  # 7

# read the memory and perform an operation
#   *** stall and forward required ***
lw $t2, 4($zero)          # 7
add $t3, $t0, $t2   # 5 + 7 = 12

# $t0 = 5, $t1 = 7, $t2 = 7, $t3 = 12 (0x0c)
```

$t2로 인해 Stall이 필요

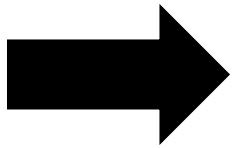
t0020-stall.fv.elf

iVerilog를 통한 Testbench 빌드

- 컴파일러를 통해 Assembly 파일을 elf 파일(실행 바이너리)로 변환

```
mips-linux-gnu-as -O0 -mips32 -o t0020-stall.fv.elf t0020-stall.fv.asm
```

mips-linux-gnu-as	-O0	-mips32	-o t0020-stall.fv.elf	t0020-stall.fv.asm
MIPS 아키텍쳐용 크로스 컴파일러	**최적화 수준**	**사용할 아키텍쳐 명**	**Output 파일명**	**Input 파일명**



t0020-stall.fv.asm(작성한 Assembly 파일)을 Input으로 하는
32bit MIPS를 최적화 진행 없이 t0020-stall.fv.elf로 컴파일을 진행해라

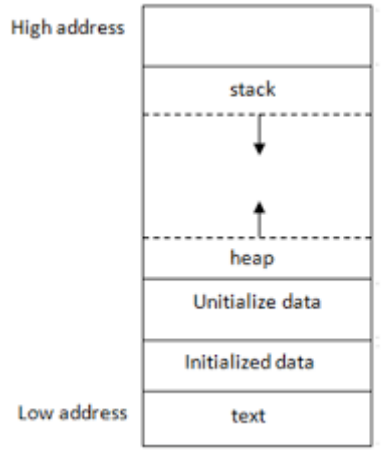
iVerilog를 통한 Testbench 빌드

- elf 파일 내부 정보 확인

readelf t0020-stall.fv.elf –a

```
ELF Header:
  Magic:   7f 45 4c 46 01 02 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, big endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              REL (Relocatable file)
  Machine:                           MIPS R3000
  Version:                           0x1
  Entry point address:               0x0
  Start of program headers:          0 (bytes into file)
  Start of section headers:          380 (bytes into file)
  Flags:                             0x50001000, o32, mips32
  Size of this header:               52 (bytes)
  Size of program headers:           0 (bytes)
  Number of program headers:         0
  Size of section headers:           40 (bytes)
  Number of section headers:         11
  Section header string table index: 10
```

```
Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .text             PROGBITS        00000000 000040 000020 00  AX  0   0 16
  [ 2] .data             PROGBITS        00000000 000060 000000 00  WA  0   0 16
  [ 3] .bss              NOBITS          00000000 000060 000000 00  WA  0   0 16
  [ 4] .reginfo          MIPS_REGINFO    00000000 000060 000018 18   A  0   0  4
  [ 5] .MIPS.abiflags    MIPS_ABIFLAGS   00000000 000078 000018 18   A  0   0  8
  [ 6] .pdr              PROGBITS        00000000 000090 000000 00      0   0  4
  [ 7] .gnu.attributes   GNU_ATTRIBUTES  00000000 000090 000010 00      0   0  1
  [ 8] .symtab           SYMTAB          00000000 0000a0 000080 10      9   8  4
  [ 9] .strtab           STRTAB          00000000 000120 000001 00      0   0  1
  [10] .shstrtab         STRTAB          00000000 000121 000059 00      0   0  1
```



iVerilog를 통한 Testbench 빌드

- elf 파일 내부의 text 정보 확인

mips-linux-gnu-objdump –d t0020-stall.fv.elf

```
# Test whether operations which require a stall and
# forward work correctly.
#

# initial values
addi $t0, $zero, 5
addi $t1, $zero, 7
addi $t2, $zero, 9

# store words in memory
sw $t0, 0($zero)  # 5
sw $t1, 4($zero)  # 7

# read the memory and perform an operation
#   *** stall and forward required ***
lw $t2, 4($zero)        # 7
add $t3, $t0, $t2   # 5 + 7 = 12

# $t0 = 5, $t1 = 7, $t2 = 7, $t3 = 12 (0x0c)
```

t0020-stall.fv.elf

```
t0020-stall.fv.elf:     file format elf32-tradbigmips


Disassembly of section .text:

00000000 <.text>:
   0:   20080005        addi    t0,zero,5
   4:   20090007        addi    t1,zero,7
   8:   200a0009        addi    t2,zero,9
   c:   ac080000        sw      t0,0(zero)
  10:   ac090004        sw      t1,4(zero)
  14:   8c0a0004        lw      t2,4(zero)
  18:   010a5820        add     t3,t0,t2
  1c:   00000000        nop
```

objdump 결과

iVerilog를 통한 Testbench 빌드

- elf 파일 내부의 text만 binary로 추출

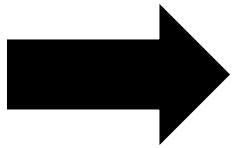
```
mips-linux-gnu-objcopy -O binary --only-section=.text t0020-stall.fv.elf t0020-stall.fv.text
```

MIPS 아키텍쳐용 객체 처리 명령어

출력 Format

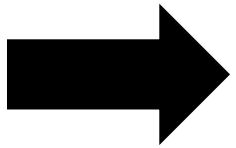
Input 파일명

Output 파일명



t0020-stall.fv.elf을 Input으로 하여
elf파일의 text 정보를 Binary로 출력해서 t0020-stall.fv.text로 저장해라

```
./bin2hex t0020-stall.fv.text > t0020-stall.fv.hex
```



저장한 text파일을 hex 파일로 변환해라

iVerilog를 통한 Testbench 빌드

- iverilog를 통한 Testbench 파일 생성

iverilog 실행	iverilog ₩
Datafile 지정	-DIM_DATA_FILE="₩"t0020-stall.fv.hex₩"" ₩
Datafile 줄 수	-DNUM_IM_DATA=`wc -l t0020-stall.fv.hex | awk '{print $1}'` ₩
Waveform 출력	-DDUMP_FILE="₩"t0020-stall.fv.vcd₩"" ₩
CPU_STAGE 지정	-DDEBUG_CPU_STAGES="1" ₩
Input들을 지정	-I../ -g2005 ₩
Output 지정	-o t0020-stall.fv ₩
Testbench 명	cpu_tb.v

iVerilog를 통한 Testbench 빌드

- 출력된 Testbench 파일을 통해 필요한 Monitor 결과만 Crop

```
./t0020-stall.fv | tail -n 1 > t0020-stall.fv.out
```

- 결과를 확인

```
vim t0020-stall.fv.out
```

```
xxxxxxxx, xxxxxxxx, 00000005, 00000007, 00000007, 0000000c, xxxxxxxx, xxxxxxxx, xxxxxxxx, xxxxxxxx
~
```

- 예상되는 결과와 동일한지 비교

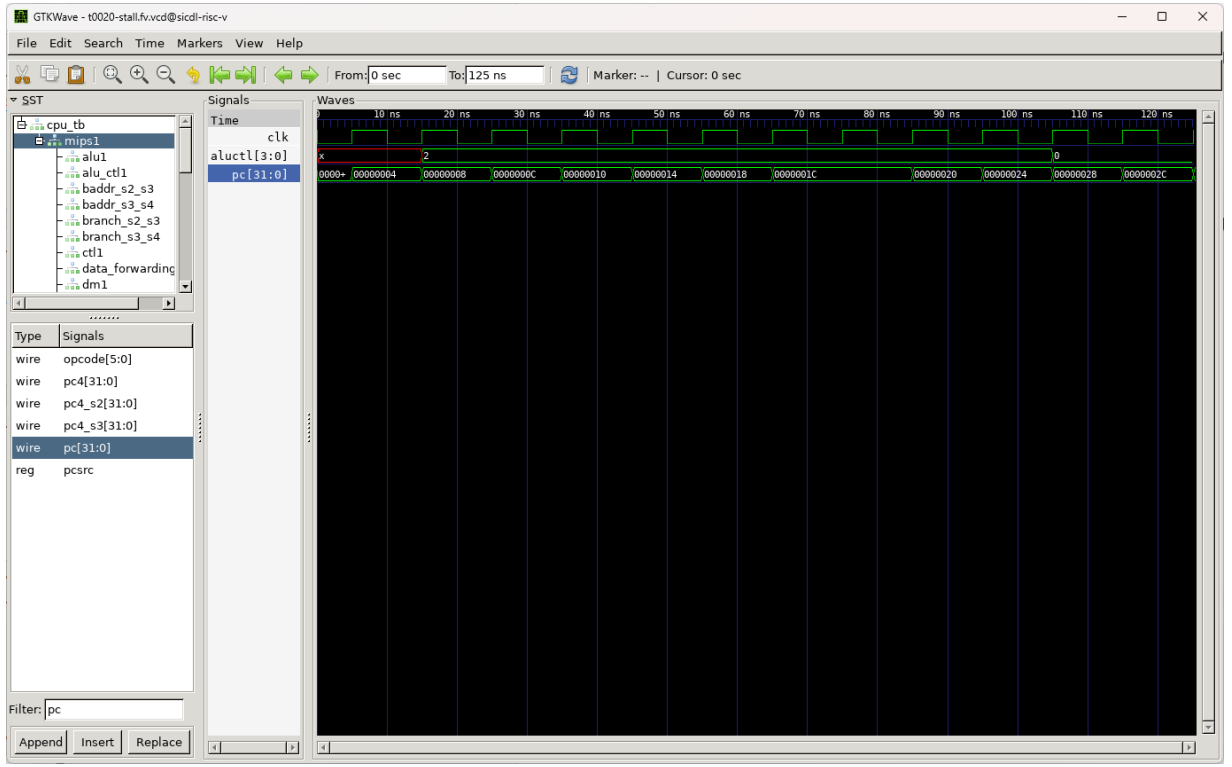
```
./check-diff.bin t0020-stall.fv.out
```

```
sicdl@sicdl-risc-v:~/mips_test/test$ ./check-diff.bin t0020-stall.fv.out
1 tests run, 1 passed, 0 failed.
```

Verification with GTKWave

- VCD Waveform을 GTKWave로 읽기

gtkwave t0020-stall.fv.vcd



Makefile 생성

- 복잡한 과정을 줄이기 위한 Makefile 생성

모든 과정은 단순히
make 하나로 처리됨.

```
mips-linux-gnu-gcc -O0 -mips32 -nostdlib -ffreestanding -o t0050-hello.elf t0050-hello.c
/usr/lib/gcc-cross/mips-linux-gnu/10/../../../../mips-linux-gnu/bin/ld: warning: cannot find entry symbol __start; defaulting to 0000000000400130
mips-linux-gnu-objcopy -O binary --only-section=.text t0050-hello.elf t0050-hello.text
./bin2hex t0050-hello.text > t0050-hello.hex
iverilog -DIM_DATA_FILE="\"t0050-hello.hex\"" \
        -DNUM_IM_DATA=`wc -l t0050-hello.hex | awk {'print $1'}` \
        -DDUMP_FILE="\"t0050-hello.vcd\"" \
        -DDEBUG_CPU_STAGES="1" \
        -I../ -g2005 \
        -o t0050-hello \
        cpu_tb.v
./t0001-final_value.fv | tail -n 1 > t0001-final_value.fv.out
./t0005-branch > t0005-branch.out
./t0005-branch.fv | tail -n 1 > t0005-branch.fv.out
./t0006-jump.fv | tail -n 1 > t0006-jump.fv.out
./t0011-operators.fv | tail -n 1 > t0011-operators.fv.out
./t0021-stall.fv | tail -n 1 > t0021-stall.fv.out
./t0030-lw_sw.fv | tail -n 1 > t0030-lw_sw.fv.out
./check-diff.bin t0001-final_value.fv.out t0005-branch.out t0005-branch.fv.out t0006-jump.fv.out t0011-operators.fv.out t0020-stall.fv.out t0021-stall.fv.out t0030-lw_sw.fv.out
8 tests run, 8 passed, 0 failed.
```

Makefile 결과

```
%.elf: %.c
	$(GCC) -O0 -mips32 -nostdlib -ffreestanding -o $@ $<

%.elf: %.asm
	$(AS) -O0 -mips32 -o $@ $<

# extract just the .text section
%.text: %.elf
	$(OBJCOPY) -O binary --only-section=.text $< $@

%.hex: %.text bin2hex
	./bin2hex $< > $@

# build the simulation executable
%: %.hex cpu_tb.v
	iverilog -DIM_DATA_FILE="\"$<\"" \
		-DNUM_IM_DATA=`wc -l $< | awk {'print $$1'}` \
		-DDUMP_FILE="\"$@.vcd\"" \
		-DDEBUG_CPU_STAGES="1" \
		-I../ -g2005 \
		-o $@ \
		cpu_tb.v

%.fv: %.fv.hex cpu_tb.v
	iverilog -DIM_DATA_FILE="\"$<\"" \
		-DNUM_IM_DATA=`wc -l $< | awk {'print $$1'}` \
		-DDUMP_FILE="\"$@.vcd\"" \
		-DDEBUG_CPU_REG="1" \
		-I../ -g2005 \
		-o $@ \
		cpu_tb.v

%.out: %
	./$< > $<.out

%.fv.out: %.fv
	./$< | tail -n 1 > $<.out

check:
	./check-diff.pl $(OUTS)
```