숭실대학교
Soongsil University

Computer Architecture

과제 #2: Architecture Simulation

2021년 1학기

Young Geun Kim (김영근)

2차 과제 목적

- Architecture Simulation을 사용한 Application 성능 평가
- Architecture Level에서 Application 특성 분석
- Application 특성을 고려한 Architecture 성능 최적화

2차 과제 내용

- **Architecture Simulation을 통한 Application 특성 및 성능 분석**
  - Architecture Simulator에서 Benchmark Application 수행
  - Architecture Level Performance Measurement를 활용한 Application 특성 분석

- **Application 특성을 고려한 Architecture 성능 최적화**
  - Architecture Simulator의 중요 Configuration 분석
  - Application 특성에 맞게 Architecture Simulator의 Configuration 조정 후 성능 향상 측정

Simulator

- **Architecture Simulator는 무엇인가?**
  - Computing Device의 Behavior를 Reproduce (재생산?) 하는 Tool

- **Simulator를 왜 사용할까?**
  - 더욱 빠르고 유연한 Software Development Cycle
    - 쉽게 Design Space Exploration을 가능하게 함
    - HW를 직접 구하지 않아도, 작성한 SW가 해당 HW에서 잘 돌아갈 수 있을지 알 수 있음
    - System 구성에 용이함

- **이번 과제에서는 SimpleScalar를 사용함**
  - Academia 및 Industry에서 널리 사용되어 왔음

SimpleScalar

Sim-Fast	Sim-Safe	Sim-Profile	Sim-Cache Sim-BPred	Sim-Outorder
-300 lines -Functional -No timing	-350 lines -Functional w/checks	-900 lines -Functional -Lots of Stats	-< 1000 lines -Functional -Cache stats -Branch stats	-3900 lines -Performance -OoO issue -Branch pred. -Mis-spec. -ALUs -Cache -TLB -200+ KIPS

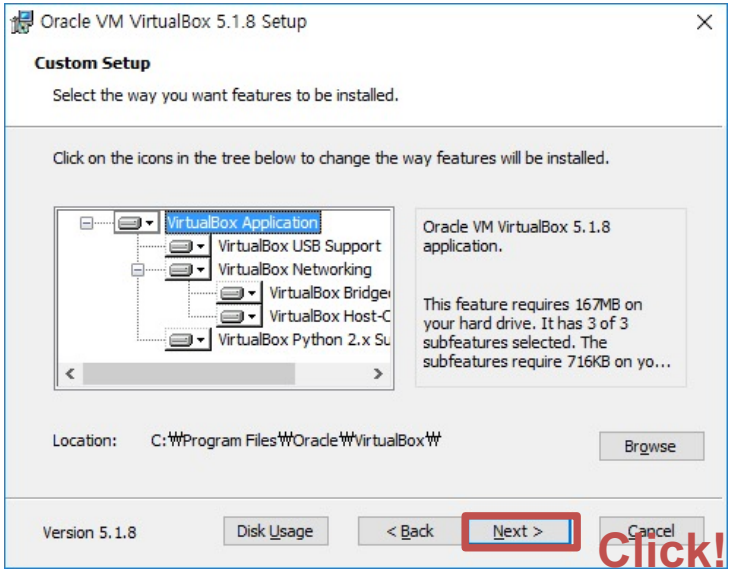
SimpleScalar 환경 설치

◆ Download VirtualBox

➢ https://virtualbox.org/wiki/Downloads

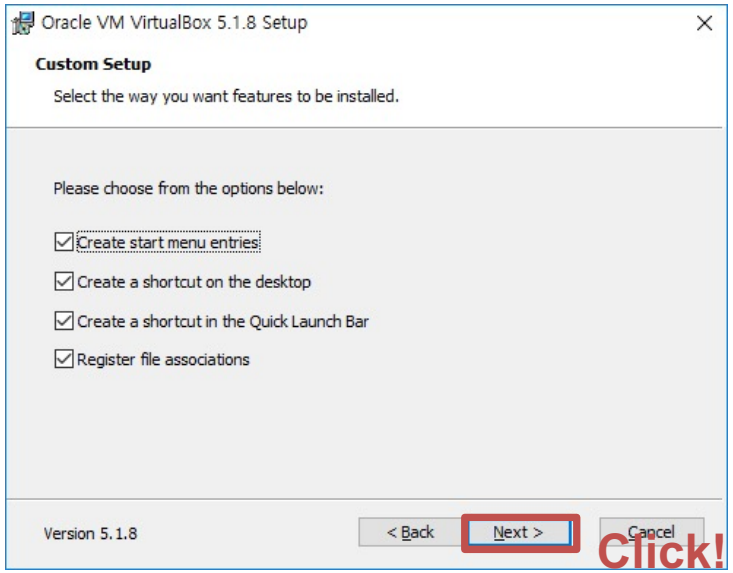
SimpleScalar 환경 설치

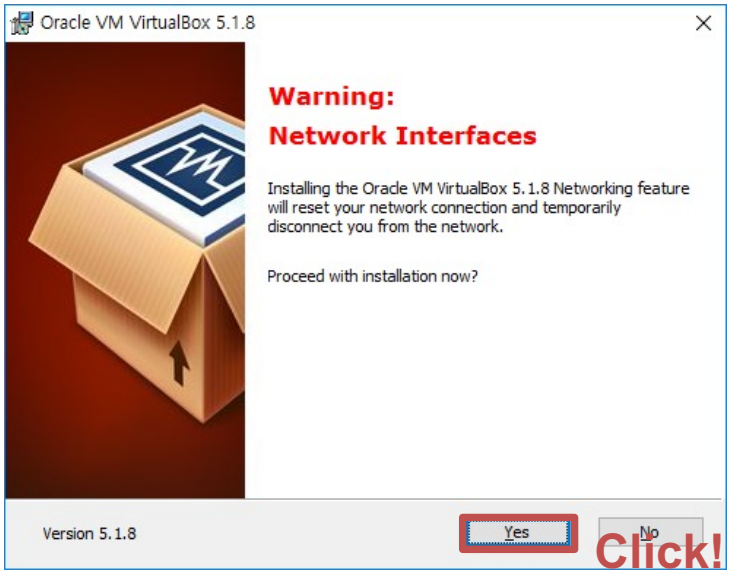
◆ Install VirtualBox



SimpleScalar 환경 설치

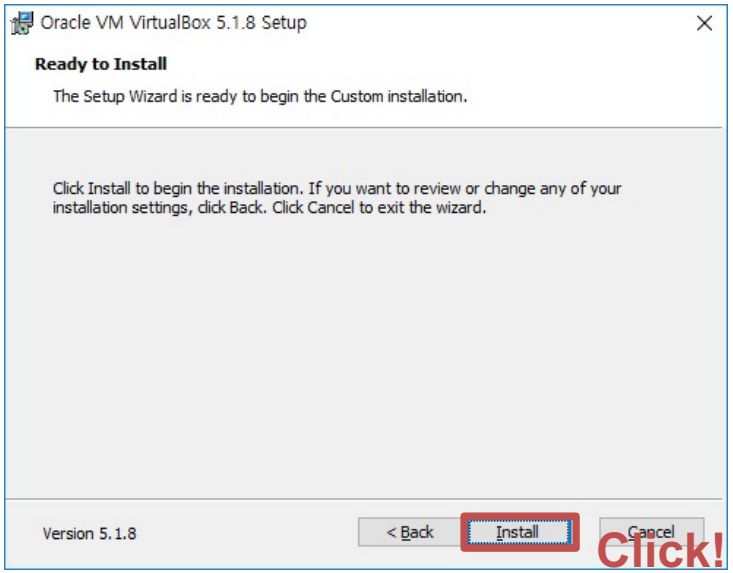
◆ Install VirtualBox

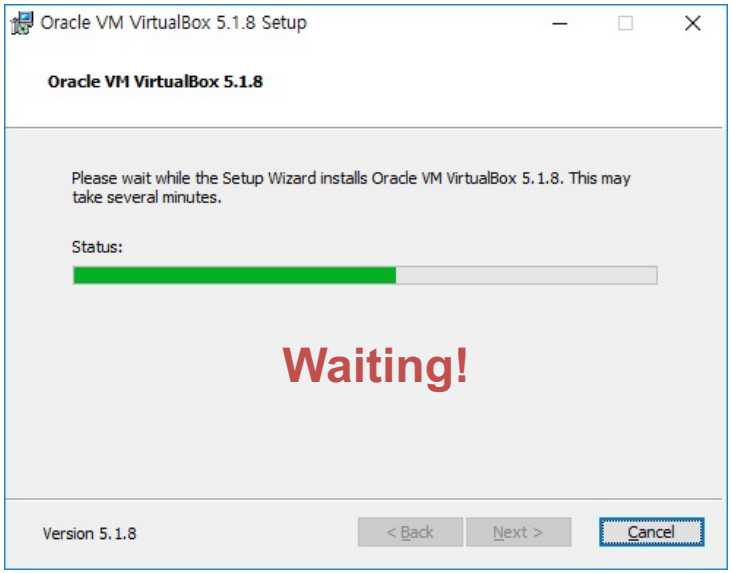




SimpleScalar 환경 설치

◆ Install VirtualBox

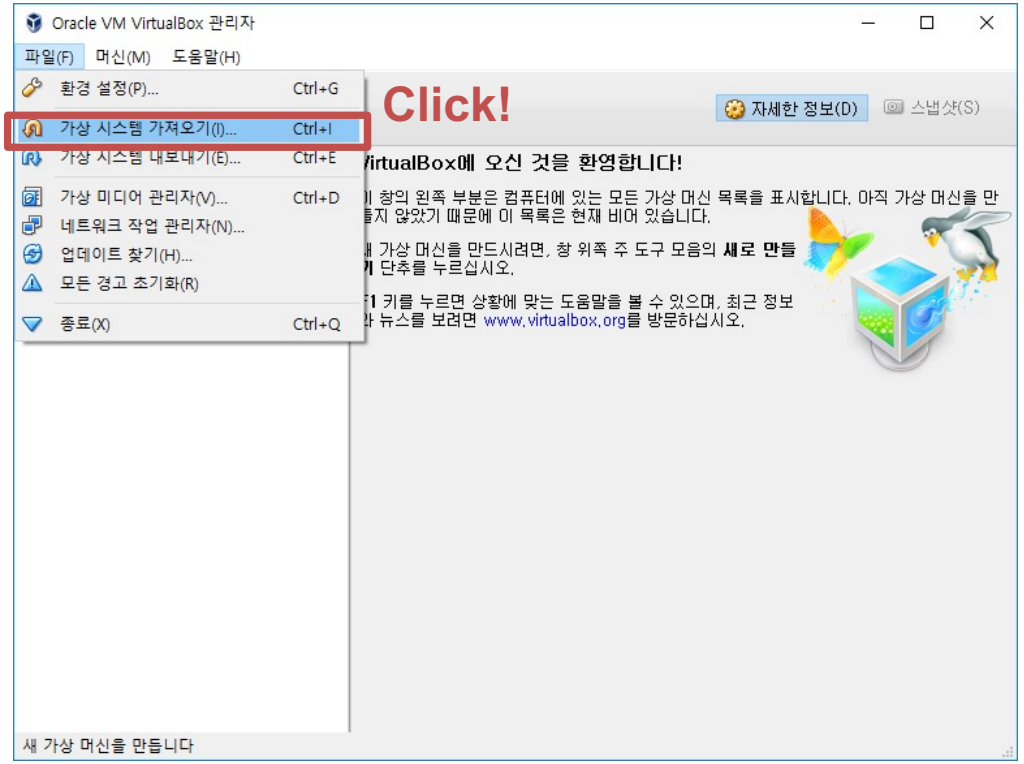




SimpleScalar 환경 설치

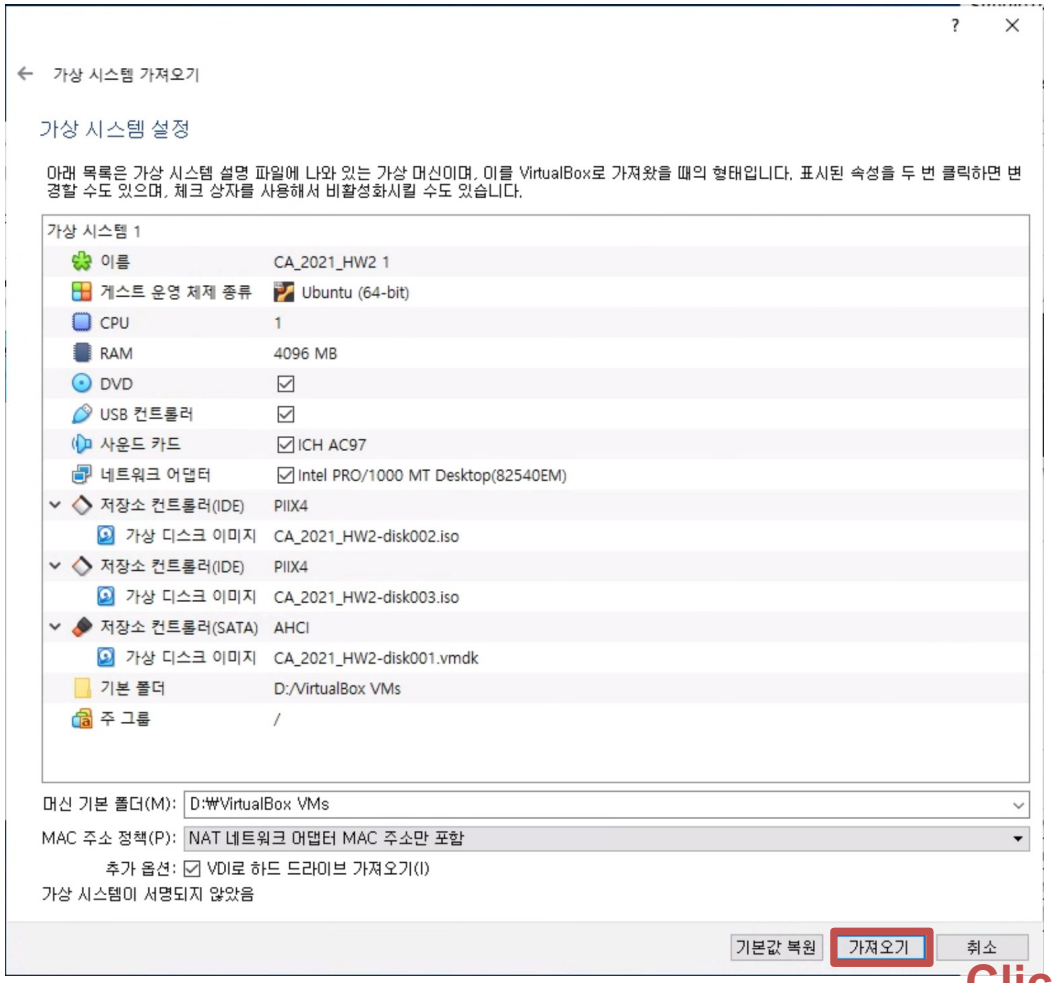
◆ Import Virtual System

➢ 스마트 캠퍼스에서 ova 파일 다운로드



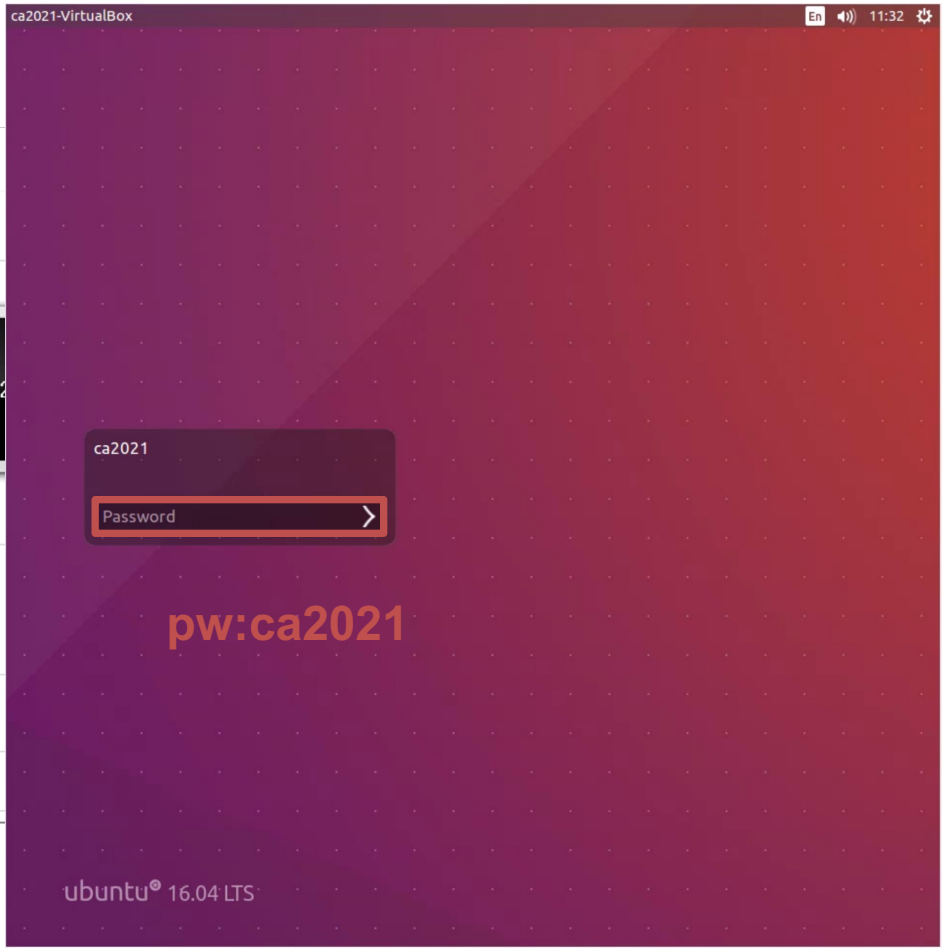
SimpleScalar 환경 설치

◆ Import Virtual System



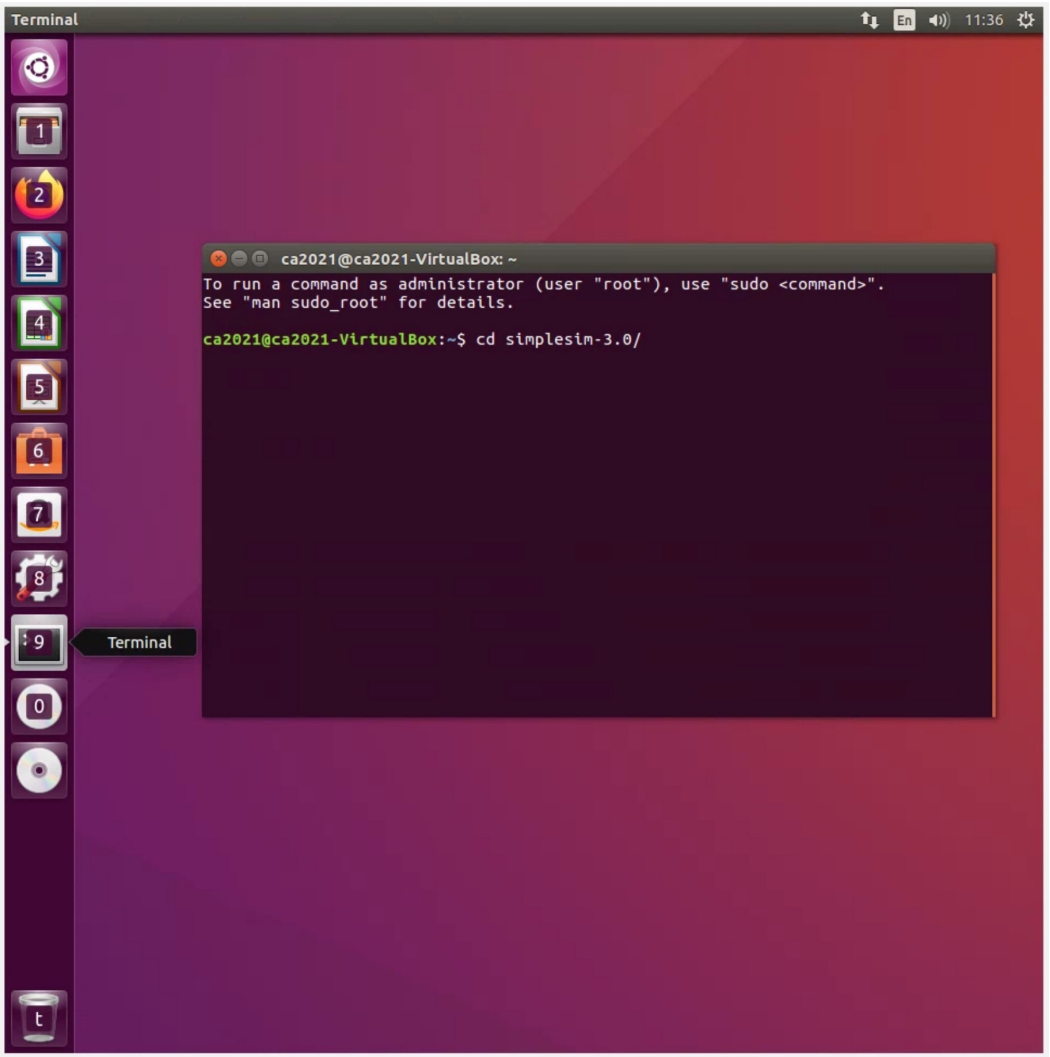
SimpleScalar 환경 설치

◆ Import Virtual System



SimpleScalar 환경 설치

◆ Execute SimpleScalar



- Execute a Terminal
- Enter the Directory of SimpleScalar

  'cd simplesim-3.0/'
- Execute Benchmarks using SimpleScalar
  - GCC

    ./sim-outorder benchmarks/cc1.alpha –O benchmarks/1stmt.i
  - ANAGRAM

    ./sim-outorder benchmarks/anagram.alpha benchmarks/words < benchmarks/anagram.in > benchmarks/OUT
  - COMPRESS95

    ./sim-outorder benchmarks/compress95.alpha < benchmarks/compress95.in > benchmarks/OUT
  - GO

    ./sim-outorder benchmarks/go.alpha 50 9 benchmarks/2stone9.in > benchmarks/OUT

SimpleScalar 환경 설치

◆ Execute SimpleScalar

```
sim: ** starting performance simulation **
warning: partially supported sigprocmask() call...

sim: ** simulation statistics **
sim_num_insn                  5019 # total number of instructions committed
sim_num_refs                  1907 # total number of loads and stores committed
sim_num_loads                  927 # total number of loads committed
sim_num_stores              980.0000 # total number of stores committed
sim_num_branches               909 # total number of branches committed
sim_elapsed_time                 1 # total simulation time in seconds
sim_inst_rate             5019.0000 # simulation speed (in insts/sec)
sim_total_insn                6034 # total number of instructions executed
sim_total_refs                2161 # total number of loads and stores executed
sim_total_loads               1091 # total number of loads executed
sim_total_stores          1070.0000 # total number of stores executed
sim_total_branches            1072 # total number of branches executed
sim_cycle                    13303 # total simulation time in cycles
sim_IPC                     0.3773 # instructions per cycle
sim_CPI                     2.6505 # cycles per instruction
sim_exec_BW                 0.4536 # total instructions (mis-spec + committed) per cycle
sim_IPB                     5.5215 # instruction per branch
IFQ_count                    11056 # cumulative IFQ occupancy
IFQ_fcount                    2360 # cumulative IFQ full count
ifq_occupancy               0.8311 # avg IFQ occupancy (insn's)
ifq_rate                    0.4536 # avg IFQ dispatch rate (insn/cycle)
ifq_latency                 1.8323 # avg IFQ occupant latency (cycle's)
ifq_full                    0.1774 # fraction of time (cycle's) IFQ was full
RUU_count                    44856 # cumulative RUU occupancy
RUU_fcount                    1039 # cumulative RUU full count
ruu_occupancy               3.3719 # avg RUU occupancy (insn's)
ruu_rate                    0.4536 # avg RUU dispatch rate (insn/cycle)
ruu_latency                 7.4339 # avg RUU occupant latency (cycle's)
ruu_full                    0.0781 # fraction of time (cycle's) RUU was full
LSQ_count                    16523 # cumulative LSQ occupancy
LSQ_fcount                     353 # cumulative LSQ full count
lsq_occupancy               1.2421 # avg LSQ occupancy (insn's)
lsq_rate                    0.4536 # avg LSQ dispatch rate (insn/cycle)
lsq_latency                 2.7383 # avg LSQ occupant latency (cycle's)
lsq_full                    0.0265 # fraction of time (cycle's) LSQ was full
sim_slip                     62260 # total number of slip cycles
avg_sim_slip                12.4049 # the average slip between issue and retirement
bpred_bimod.lookups            1115 # total number of bpred lookups
bpred_bimod.updates             909 # total number of updates
bpred_bimod.addr_hits           503 # total number of address-predicted hits
bpred_bimod.dir_hits            730 # total number of direction-predicted hits (includes addr-hits)
bpred_bimod.misses              179 # total number of misses
bpred_bimod.jr_hits             108 # total number of address-predicted hits for JR's
bpred_bimod.jr_seen             182 # total number of JR's seen
bpred_bimod.jr_non_ras_hits.PP           16 # total number of address-predicted hits for non-RAS JR's
bpred_bimod.jr_non_ras_seen.PP           38 # total number of non-RAS JR's seen
```

- **Simulation Statistics**
  - **Total number of instructions**
  - **Total number of loads and stores**
  - **Total number of branches**
  - **Total simulation time in cycles**
  - **Instruction per cycle**
  - **Cycles per instruction**

SimpleScalar 환경 설치

◆ Configurations of Simulation

```
ca2021@ca2021-VirtualBox:~/simplesim-3.0$ ./sim-outorder -h
sim-outorder: SimpleScalar/Alpha Tool Set version 3.0 of August, 2003.
Copyright (c) 1994-2003 by Todd M. Austin, Ph.D. and SimpleScalar, LLC.
All Rights Reserved. This version of SimpleScalar is licensed for academic
non-commercial use.  No portion of this work may be used by any commercial
entity, or for any commercial purpose, without the prior written permission
of SimpleScalar, LLC (info@simplescalar.com).

Usage: ./sim-outorder {-options} executable {arguments}

sim-outorder: This simulator implements a very detailed out-of-order issue
superscalar processor with a two-level memory system and speculative
execution support.  This simulator is a performance simulator, tracking the
latency of all pipeline operations.

#
# -option        <args>           #    <default> # description
#
-config          <string>         #       <null> # load configuration from a file
-dumpconfig      <string>         #       <null> # dump configuration to a file
-h               <true|false>     #         true # print help message
-v               <true|false>     #        false # verbose operation
-d               <true|false>     #        false # enable debug message
-i               <true|false>     #        false # start in Dlite debugger
-seed            <int>            #            1 # random number generator seed (0 for timer seed
)
-q               <true|false>     #        false # initialize and terminate immediately
-chkpt           <string>         #       <null> # restore EIO trace execution from <fname>
-redir:sim       <string>         #       <null> # redirect simulator output to file (non-interac
tive only)
-redir:prog      <string>         #       <null> # redirect simulated program output to file
-nice            <int>            #            0 # simulator scheduling priority
-max:inst        <uint>           #            0 # maximum number of inst's to execute
-fastfwd         <int>            #            0 # number of insts skipped before timing starts
-ptrace          <string list...> #       <null> # generate pipetrace, i.e., <fname|stdout|stderr
> <range>
-fetch:ifqsize   <int>            #            4 # instruction fetch queue size (in insts)
-fetch:mplat     <int>            #            3 # extra branch mis-prediction latency
-fetch:speed     <int>            #            1 # speed of front-end of machine relative to exec
ution core
-bpred           <string>         #        bimod # branch predictor type {nottaken|taken|perfect|
bimod|2lev|comb}
-bpred:bimod     <int>            # 2048 # bimodal predictor config (<table size>)
-bpred:2lev      <int list...>    # 1 1024 8 0 # 2-level predictor config (<l1size> <l2size> <his
t_size> <xor>)
-bpred:comb      <int>            # 1024 # combining predictor config (<meta_table_size>)
-bpred:ras       <int>            #            8 # return address stack size (0 for no return sta
ck)
-bpred:btb       <int list...>    # 512 4 # BTB config (<num_sets> <associativity>)
-bpred:spec_update <string>         #       <null> # speculative predictors update in {ID|WB} (de
fault non-spec)
-decode:width    <int>            #            4 # instruction decode B/W (insts/cycle)
-issue:width     <int>            #            4 # instruction issue B/W (insts/cycle)
-issue:inorder   <true|false>     #        false # run pipeline with in-order issue
-issue:wrongpath <true|false>     #         true # issue instructions down wrong execution paths
```

- Type ./sim-outorder -h
- Important Configurations
  - fetch:ifqsize
  - bpred
  - decode:width
  - issue:width
  - cache:dl1
  - cache:dl2
  - cache:il1
  - cache:il2
  - res:ialu
  - res:imult
  - res:memport
  - res:fpalu
  - res:fpmult

주의: cache:dl1lat, cache:dl2lat, 등 Latency 값을 직접 변경해서는 안됨

과제 채점 Focus

- **과제 Report의 비중: 100%**
  - Report에 반드시 포함되어야 하는 내용
    - 학과, 학번, 이름, 제출 일자
    - 각 Benchmark Application에 대해
      - 기본 Configuration에서 측정한 성능 기술 **(2점)**
      - 각 Application이 어떤 특성을 갖고 있는지 분석한 내용 서술 **(2점)**
    - SimpleScalar Simulator Configuration 분석한 내용 서술 **(2점)**
    - 각 Benchmark Application에 대해
      - Configuration 변경하고, 측정한 성능 기술 **(2점)**
      - 왜 Configuration을 그렇게 변경했는지 기술 **(4점)**
      - Configuration 변경 전과 변경 후 성능 Graph로 비교 **(3점)**

- **성능 향상 수치 (4개 Application의 평균)의 상위 10명은 추가 점수 부여 (1점)**

과제 채점 Focus

- **제출 기한**
  - 6월 11일 (금) 11:59 PM

- **제출 방법**
  - 스마트 캠퍼스를 통해 파일들 제출
  - 제출할 파일 목록
    - 보고서
  - 파일 제출 방법
    - "ca2_학번.zip"으로 압축하여 제출
      (다른 형식도 무관)