TA Session 1

조교 김진표, 박대영, 신준식, 정재훈

pp-ta@aces.snu.ac.kr

구성

- 실습 과제 1
  - 1. 실습 서버 스펙 확인 (CPU, GPU, Memory)
  - 2. CPU 의 실수 연산 성능 (Rpeak) 계산
  - 3. 부동 소수점 표현

1. 실습 서버 스펙 확인

1-1. 다음 중 계산 노드당 장착된 CPU 의 종류와 개수로 옳은 것은?

a. AMD EPYC 7452 32-Core Processor x1

b. AMD EPYC 7702P 64-Core Processor x1

c. Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz x1

d. Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz x2

e. Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz x1

f. Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz x2

1. 실습 서버 스펙 확인

1-2. 다음 중 계산 노드당 장착된 GPU 의 종류와 개수로 옳은 것은?

a. NVIDIA GeForce RTX 3090 x4

b. NVIDIA GeForce RTX 3090 x2

c. NVIDIA GeForce RTX 3090 x1

d. NVIDIA GeForce RTX 3080 x1

e. Tesla V100-PCIE-32GB x4

f. Tesla V100-PCIE-32GB x1

1. 실습 서버 스펙 확인

1-3. 다음 중 계산 노드당 메모리 용량으로 옳은 것은?

가장 근접한 답안을 선택

a. 16GB

b. 32GB

c. 64GB

d. 128GB

e. 256GB

f. 512GB

1. 실습 서버 스펙 확인

- 제출 방법
  - ~/submit/HW1/prob1.txt 파일 생성
  - 파일 내용은 정확히 3글자. 각 소문제의 답안을 나타냄.

- 아래 명령으로 체크 (아래 예시는 정답이 아님!)

```
kjp4155@login:~$ cat ~/submit/HW1/prob1.txt
abd
kjp4155@login:~$
```

2. CPU 의 실수 연산 성능 계산

- 계산 노드에 장착된 CPU의 Theoretical Peak FLOPS 를 계산하기
  - 1개 이상의 CPU 가 장착되어 있어도, 1개 기준으로 계산
  - FP32 (32-bit float) 기준으로 계산
  - CPU 1개의 모든 자원 (core, 연산 유닛 등)을 최대한 사용한 경우로 계산
  - CPU clock 은 2.10 GHz 기준으로 계산
- 제출 방법
  - ~/submit/HW1/prob2.txt 파일 생성
  - 파일 내용은 CPU의 TFLOPS 를 소숫점 2자리까지 (그 밑은 반올림)
  - 아래 예시는 정답이 아님!

```
kjp4155@login:~$ cat ~/submit/HW1/prob2.txt
1.12
kjp4155@login:~$
```

3. 수의 이진 표현

- 다양한 타입의 수를 입력받아 이진 표현을 출력하는 프로그램을 작성
  - `int, long, float, double` 의 타입
  - main.cpp, Makefile, sample input 이 `/skeleton/HW1` 디렉토리에 주어짐.
  - main.cpp 의 TODO 를 작성할 것. 다른 부분은 수정하지 않을 것
  - make run 으로 프로그램을 테스트 해 볼 수 있음

```
kjp4155@login:~$ cp -r /skeleton/HW1/ ~/
kjp4155@login:~$ ls HW1
main.cpp  Makefile  sample.in
kjp4155@login:~$ cd HW1
kjp4155@login:~/HW1$ make run
g++ -o hw1 main.cpp
./hw1 < sample.in
kjp4155@login:~/HW1$
```

3. 수의 이진 표현

- 다양한 타입의 수를 입력받아 이진 표현을 출력하는 프로그램을 작성
  - 아래 예시는 정확히 구현했을 경우의 출력임
  - 로그인 노드에서 작업 및 실행

```
kjp4155@login:~/HW1$ ls
main.cpp  Makefile  sample.in
kjp4155@login:~/HW1$ cat sample.in
-1
99
103.625
-202.332
kjp4155@login:~/HW1$ make run
g++ -o hw1 main.cpp
./hw1 < sample.in
11111111111111111111111111111111
00000000000000000000000000000000000000000000000000000001100011
01000010110011110100000000000000
11000000011010010100101010011111101111100111011011001000101101000
kjp4155@login:~/HW1$ ls
hw1  main.cpp  Makefile  sample.in
kjp4155@login:~/HW1$
```

3. 수의 이진 표현

- 제출 방법
  - ~/submit/HW1/main.cpp 로 구현이 완료된 main.cpp 를 복사
  - Makefile, sample.in 등 다른 파일은 제출할 필요 없음

```
kjp4155@login:~/HW1$ ls
hw1  main.cpp  Makefile  sample.in
kjp4155@login:~/HW1$ cp main.cpp ~/submit/HW1/main.cpp
kjp4155@login:~/HW1$ head ~/submit/HW1/main.cpp
#include <iostream>

void printIntAsBinary(int i)
{
```

제출 방법 정리

- 각자의 홈 디렉토리에 `~/submit/HW1` 디렉토리 생성
- 1, 2, 3 번 과제의 답안이 적힌 파일을 각각 다음과 같은 이름으로 제출
  - prob1.txt
  - prob2.txt
  - main.cpp

- **디렉토리명, 파일명이 정확히 일치하도록**
- 즉, `ls` 명령어의 출력이 아래 스크린샷과 같이 나와야 함

```
kjp4155@login:~$ ls ~/submit/HW1
main.cpp  prob1.txt  prob2.txt
kjp4155@login:~$ ls -l ~/submit/HW1
total 12
-rw-r--r-- 1 kjp4155 user 1492 Apr 19 06:00 main.cpp
-rw-r--r-- 1 kjp4155 user    4 Apr 19 05:39 prob1.txt
-rw-r--r-- 1 kjp4155 user    5 Apr 19 05:41 prob2.txt
kjp4155@login:~$
```

잘못된 제출 예시

- 잘못된 제출 디렉토리
  - **[O] ~/submit/HW1**
  - [X] ~/submit/HW_1
  - [X] ~/submit/hw1
  - [X] ~/submit/hw
  - [X] ~/submit/homework1
  - [X] ~/submit/assignment
  - [X] ~/submit/Hw1
  - [X] ~/submit/hw1/hw1
  - [X] ~/HW1
  - [X] ~/submt/HW1
  - [X] ~/HW1/HW1

잘못된 제출 예시

- 잘못된 제출 파일명
  - **[O] ~/submit/HW1/prob1.txt**
  - [X] ~/submit/HW1/prob1
  - [X] ~/submit/HW1/prob_1.txt
  - [X] ~/submit/HW1/probA.txt
  - [X] ~/submit/HW1/Prob1.txt
  - [X] ~/submit/HW1/prob1.ans
  - [X] ~/submit/HW1/ans1.txt
  - [X] ~/submit/HW1/porb1.txt
  - [X] ~/submit/HW1/PROB1.txt
  - [X] ~/submit/HW1/1.txt
  - [X] ~/submit/HW1/1.py
  - [X] ~/submit/HW1/A.txt

잘못된 제출 예시

- 불필요한 / 잘못된 출력

```
kjp4155@login:~/HW1$ make run
./hw1 < sample.in
11111111111111111111111111111111
00000000000000000000000000000000000000000000000000000001100011
01000010110011110100000000000000
11000000011010010100101010011111101111100111011011001000101101000
kjp4155@login:~/HW1$
```

Correct

```
kjp4155@login:~/HW1$ make run
g++ -o hw1 main.cpp
./hw1 < sample.in
Int: 11111111111111111111111111111111
Long: 00000000000000000000000000000000000000000000000000000001100011
Float: 01000010110011110100000000000000
Double: 11000000011010010100101010011111101111100111011011001000101101000
```

Wrong!

주의사항

- **제출 기한: 4월 22일 (금) 08:59**
  - 기한이 되면 프로그램이 자동으로 제출 파일을 복사해 감
  - 파일명, 디렉토리, 형식을 정확히 지켰는지 체크

- **조교가 실수를 수정해줄 것이라 기대하지 않을 것**
  - 제출 디렉토리 확인
  - 제출 파일명 확인
  - 제출한 파일 내용 확인
  - 프로그램이 필요없는 출력을 하는 지 확인
  - 디버깅 코드 등 불필요한 코드를 모두 제거했는지 확인

과제1 - 해설

- 제출이 잘 되었는지 체크하는 유틸리티를 제공함
- 로그인 노드에서 `check-submit` 명령으로 실행

```
kjp4155@login:~$ ls -al submit/HW1/
total 12
drwxr-xr-x 2 kjp4155 user 4096 Apr 20 18:52 .
drwxr-xr-x 3 kjp4155 user 4096 Apr 19 05:00 ..
-rw-r--r-- 1 kjp4155 user    0 Apr 20 18:51 main.cpp
-rw-r--r-- 1 kjp4155 user    5 Apr 20 18:52 prob2.txt
kjp4155@login:~$ check-submit
Username: kjp4155
------------------------------------
HW1 prob1.txt   : X file not found
HW1 prob2.txt   : OK
HW1 main.cpp    : X empty file
------------------------------------
kjp4155@login:~$
```

과제1 - 해설

1-1. 다음 중 계산 노드당 장착된 CPU 의 종류와 개수로 옳은 것은?

f. Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz x2

```
kjp4155@login:~$ srun -N 1 lscpu
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   46 bits physical, 48 bits virtual
CPU(s):                          80
On-line CPU(s) list:             0-79
Thread(s) per core:              2
Core(s) per socket:              20
Socket(s):                       2
NUMA node(s):                    2
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           85
Model name:                      Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz
Stepping:                        7
CPU MHz:                         800.899
CPU max MHz:                     3900.0000
CPU min MHz:                     800.0000
BogoMIPS:                        4200.00
Virtualization:                  VT-x
L1d cache:                       1.3 MiB
```

과제1 - 해설

1-2. 다음 중 계산 노드당 장착된 GPU 의 종류와 개수로 옳은 것은?

a. NVIDIA GeForce RTX 3090 x4

```
kjp4155@login:~$ srun -N 1 nvidia-smi -q | grep Name
    Product Name                          : NVIDIA GeForce RTX 3090
    Product Name                          : NVIDIA GeForce RTX 3090
    Product Name                          : NVIDIA GeForce RTX 3090
    Product Name                          : NVIDIA GeForce RTX 3090
kjp4155@login:~$
```

과제1 - 해설

1-3. 다음 중 계산 노드당 메모리 용량으로 옳은 것은?

가장 근접한 답안을 선택

e. 256GB

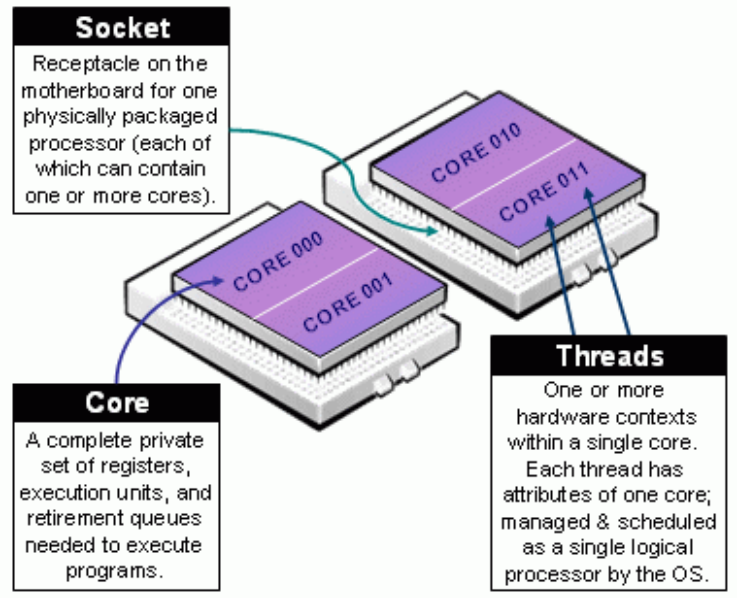
```
kjp4155@login:~$ srun -N 1 free -h
               total        used        free      shared  buff/cache   available
Mem:           251Gi       1.6Gi       248Gi       3.0Mi       1.0Gi       248Gi
Swap:          8.0Gi          0B       8.0Gi
kjp4155@login:~$
```

과제1 - 해설

2. 실습 서버 계산 노드에 장착된 CPU 1개의 Theoretical peak FLOPS 를 계산하기

- Theoretical peak FLOPS = 이론상 최대 연산 성능

(Clock frequency) x (# of cores) x (# instruction / cycle) x (# op / instruction)



과제1 - 해설

2. 실습 서버 계산 노드에 장착된 CPU 1개의 Theoretical peak FLOPS 를 계산하기

- Theoretical peak FLOPS = 이론상 최대 연산 성능

(Clock frequency) x (# of cores) x (# instruction / cycle) x (# op / instruction)

- Clock frequency = 2.10 Ghz
- # of cores = 20 (Physical core 개수)
- # instruction / cycle = 2 (AVX512)
- # op / instruction = 32 (AVX512, FMA 사용 = 16회 곱셈 + 16회 덧셈)
- => 2.688 ~= 2.69 FP32 TFLOPS

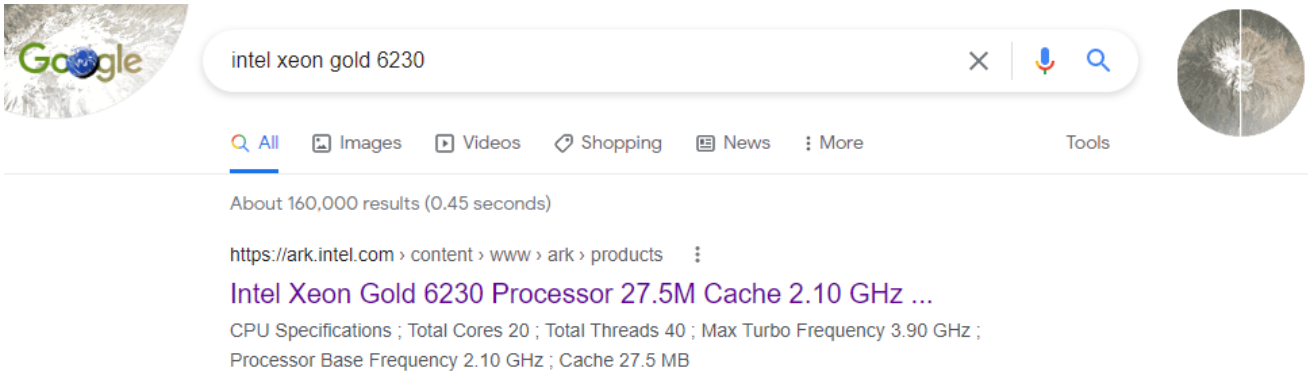
과제1 - 해설

- Clock frequency = 2.10 Ghz
- # of cores = 20 (Physical core 개수)

```
kjp4155@login:~$ srun -N 1 lscpu
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   46 bits physical, 48 bits virtual
CPU(s):                          80
On-line CPU(s) list:             0-79
Thread(s) per core:              2
Core(s) per socket:              20
Socket(s):                       2
NUMA node(s):                    2
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           85
Model name:                      Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz
Stepping:                        7
CPU MHz:                         800.899
CPU max MHz:                     3900.0000
CPU min MHz:                     800.0000
BogoMIPS:                        4200.00
Virtualization:                  VT-x
L1d cache:                       1.3 MiB
```

과제1 - 해설

- # instruction / cycle = 2 (AVX512)
- # op / instruction = 32 (AVX512, FMA 사용 = 16회 곱셈 + 16회 덧셈)



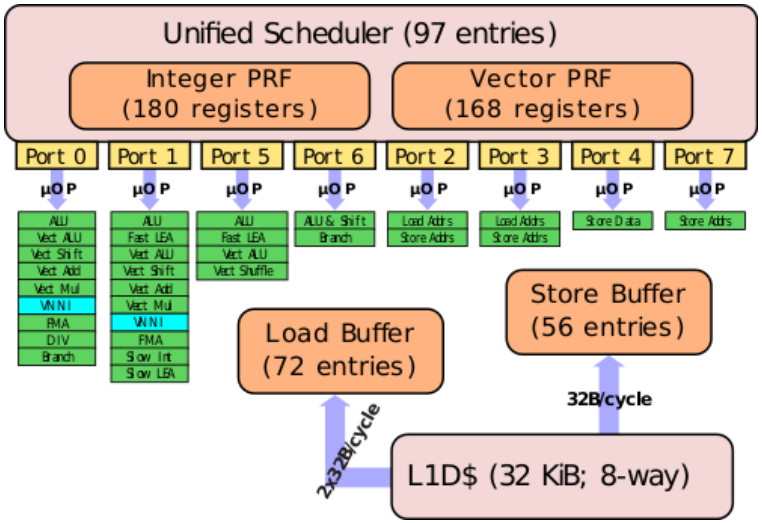
Instruction Set Extensions ?	Intel® SSE4.2, Intel® AVX, Intel® AVX2, Intel® AVX-512
# of AVX-512 FMA Units ?	2

https://ark.intel.com/content/www/us/en/ark/products/192437/intel-xeon-gold-6230-processor-27-5m-cache-2-10-ghz.html

과제1 - 해설

- # instruction / cycle = 2 (AVX512)
- # op / instruction = 32 (AVX512, FMA 사용 = 16회 곱셈 + 16회 덧셈)

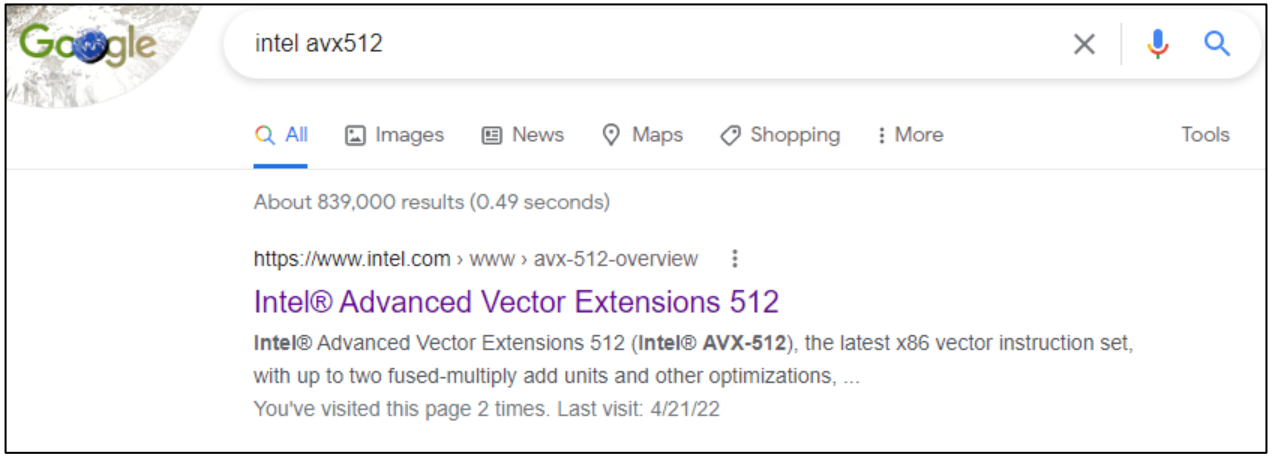
- The latency and throughput of floating point ADD, MUL, and FMA were made uniform at 4 cycles with a throughput of 2 µOPs/clock.



https://en.wikichip.org/wiki/intel/microarchitectures/cascade_lake
https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(client)#Execution_engine

과제1 - 해설

- # instruction / cycle = 2 (AVX512)
- # op / instruction = 32 (AVX512, FMA 사용 = 16회 곱셈 + 16회 덧셈)



Enhanced Vector Processing Capabilities

With ultra-wide 512-bit vector operations capabilities, Intel® AVX-512 can handle your most demanding computational tasks.

Applications can pack 32 double precision and 64 single precision floating point operations per clock cycle within the 512-bit vectors, as well as eight 64-bit and sixteen 32-bit integers, with up to two 512-bit fused-multiply add (FMA) units, thus doubling the width of data registers, doubling the number of registers, and, doubling the width of FMA units, compared to Intel® Advanced Vector Extensions 2 (Intel® AVX2).[2 3]

과제1 - 해설

3. 다양한 타입의 수를 입력받아 이진 표현을 출력하는 프로그램을 작성

- 다양한 풀이가 가능
  - 직접 이진 표현 변환 알고리즘을 구현
  - 변수에 저장된 값을 이진 표현으로 출력 (Union, casting 등..)

- 예시 답안을 /skeleton/HW1-ans 디렉토리에 제공함

```
void printDoubleAsBinary(double d)
{
        using type_t = double;
        using cast_type_t = unsigned long;
        cast_type_t v = *((cast_type_t*)(&d));
        unsigned int nbits = sizeof(type_t)*8;
        for (int i = 0; i < nbits; i++) {
                std::cout << ((v & (1ul << (nbits-1-i))) >> (nbits-1-i));
        }
        std::cout << "\n";
}
```

SMT 와 OS process

- SMT: hardware multithreading (a.k.a. Intel’s Hyperthreading)
  - OS 에서 관리하는 프로세스와는 관련이 없음

- CPU (Physical core) 입장에서는 OS 가 관리하는 프로세스 중 어떤 것을 실행하는지 중요치 않음

- 같은 physical core 의 두 logical core 가 서로 다른 프로세스의 스레드를 실행 가능