C++ Korea 4th Seminar

C++ 프로젝트 ~처음 만난 세계~

Optimize C++

하드웨어 특성을 고려한 최적화 접근

Orichal Partners Korea

박재현

Overview

1. 최적화란?
2. 일반적 접근방식
3. 최적화에 관계된 하드웨어 특성들
4. C++ 와의 연관
5. Optimize C++
6. Tool 활용

Goals

최적화가 무엇인지 이해
최적화 이슈를 어떻게 접근하는지 이해

최적화란?

사전적 의미 op•ti•mize (ˈɒp təˌmaɪz)

v. **-mized, -miz•ing.** *v.t.*

1. to make as effective, perfect, or useful as possible.

2. to make the best of.

3. to write or rewrite (the instructions in a computer program) for maximum efficiency and speed in retrieval, storage, or execution.

4. *Math.* to determine the maximum or minimum values of (a specified function that is subject to a set of constraints).

*v.i.***5.** to be optimistic.

최적-화 (最適-化) : 가장 알맞다

잘못된 상식

- 하드웨어의 성능이 증가하고 있으므로 문제가 아니다
- 가독성이 떨어지며 더 어려운 소프트웨어를 의미
- 추가적인 유지보수의 부담을 초래
- 하드웨어를 깊이 이해해야 한다

현실은?

최적화는...

- 필요할 때만 한다
- 필요한 만큼만 한다
- 필요한 위치에만 한다
- 적절한 방법으로 한다
- 때로는 눈속임(?)도 한다

대상은?

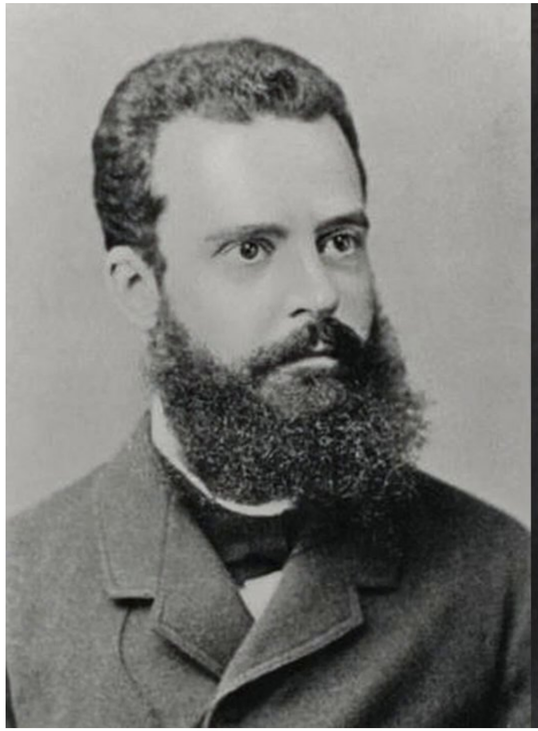
- 속도의 개선
- 공간의 절약
- 개발업무 편의

일반적 접근방식

Feat. 프로파일링

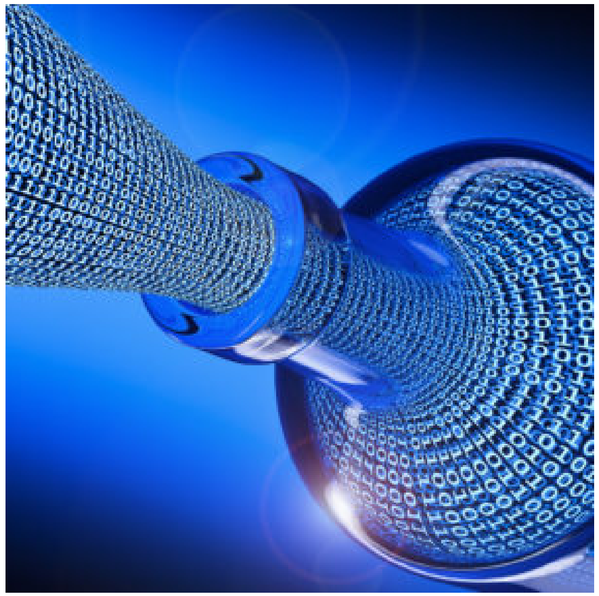
20:80

20%의 코드가 80%의 수행 시간을 차지한다.
Bottleneck 검출



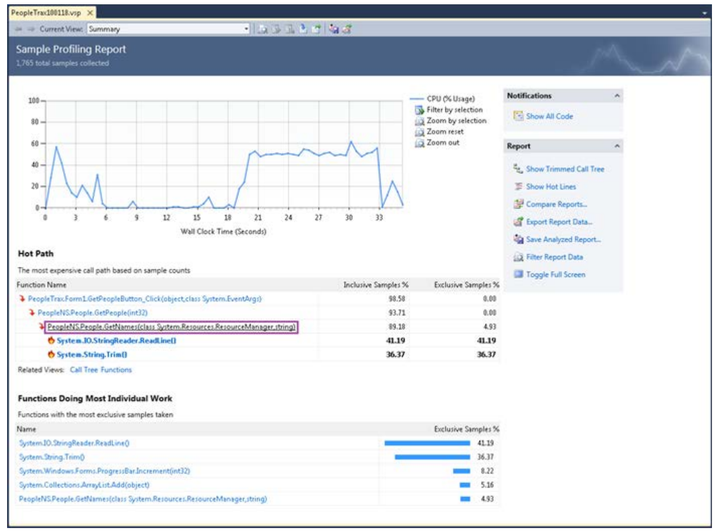
접근

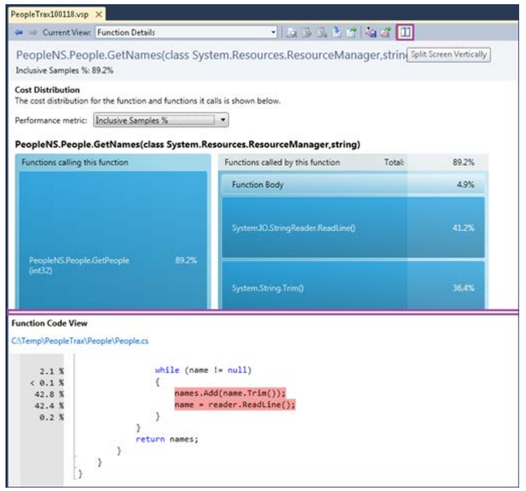
- 필요할 때 → 성능이 문제가 되는가?
- 필요한 만큼 → 최적화 목표를 설정하고
- 필요한 위치에 → 병목구간 검출
- 적절한 방법으로 → 방법은 많다
- 때로는 눈속임(?)도 한다 → 체감이 중요

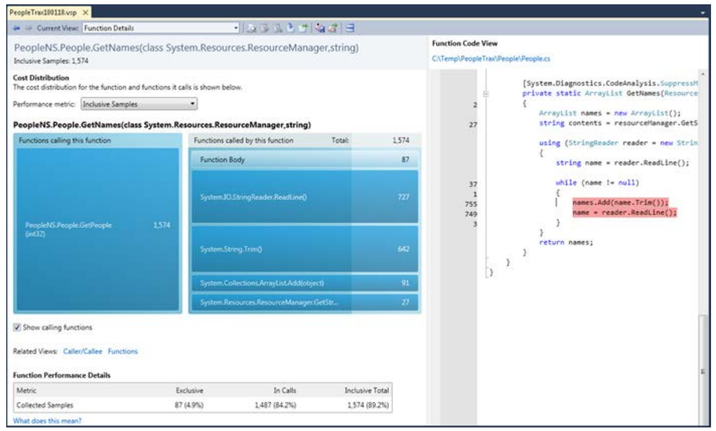


병목구간 검출

Profiling

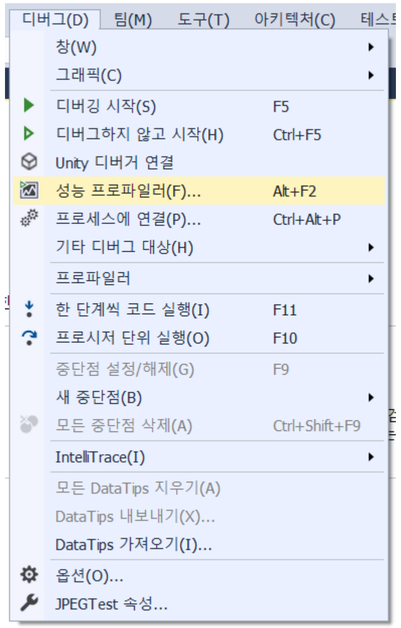


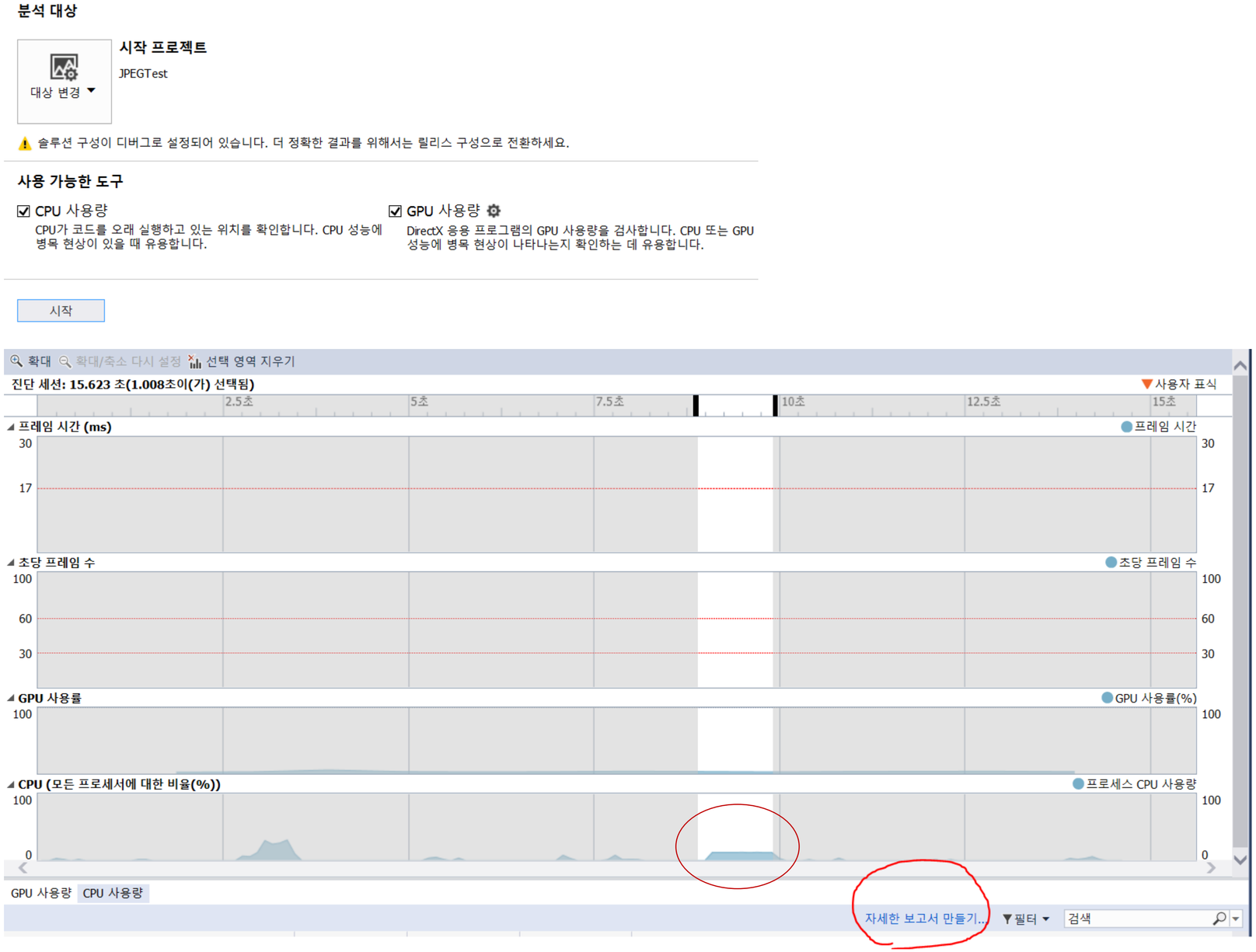




Tool 활용 – 성능 프로파일러

성능 프로파일러

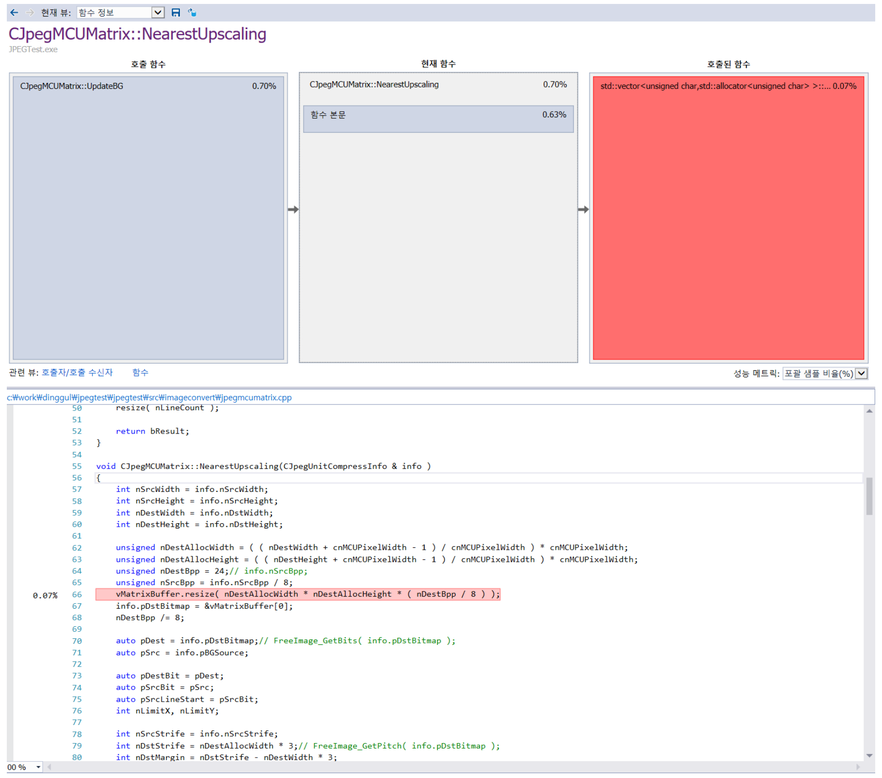

디버그(D)
팀(M)
도구(T)
아키텍처(C)
창(W)
그래픽(C)
디버깅 시작(S) F5
디버그하지 않고 시작(H) Ctrl+F5
Unity 디버거 연결
성능 프로파일러(F)... Alt+F2
프로세스에 연결(P)... Ctrl+Alt+P
기타 디버그 대상(H)
프로파일러
한 단계씩 코드 실행(I) F11
프로시저 단위 실행(O) F10
중단점 설정/해제(G) F9
새 중단점(B)
모든 중단점 삭제(A) Ctrl+Shift+F9
IntelliTrace(I)
모든 DataTips 지우기(A)
DataTips 내보내기(X)...
DataTips 가져오기(I)...
옵션(O)...
JPEGTest 속성...


분석 대상
시작 프로젝트
JPEGTest
대상 변경
솔루션 구성이 디버그로 설정되어 있습니다. 더 정확한 결과를 위해서는 릴리스 구성으로 전환하세요.
사용 가능한 도구
CPU 사용량
CPU가 코드를 오래 실행하고 있는 위치를 확인합니다. CPU 성능에 병목 현상이 있을 때 유용합니다.
GPU 사용량
DirectX 응용 프로그램의 GPU 사용량을 검사합니다. CPU 또는 GPU 성능에 병목 현상이 나타나는지 확인하는 데 유용합니다.
시작
확대
확대/축소 다시 설정
선택 영역 지우기
진단 세션: 15.623 초(1.008초이(가) 선택됨)
사용자 표식
2.5초
5초
7.5초
10초
12.5초
15초
프레임 시간 (ms)
프레임 시간
30
17
초당 프레임 수
초당 프레임 수
100
60
30
GPU 사용률
GPU 사용률(%)
CPU (모든 프로세서에 대한 비율(%))
프로세스 CPU 사용량
0
GPU 사용량
CPU 사용량
자세한 보고서 만들기...
필터
검색

성능 프로파일러

현재 뷰: 함수

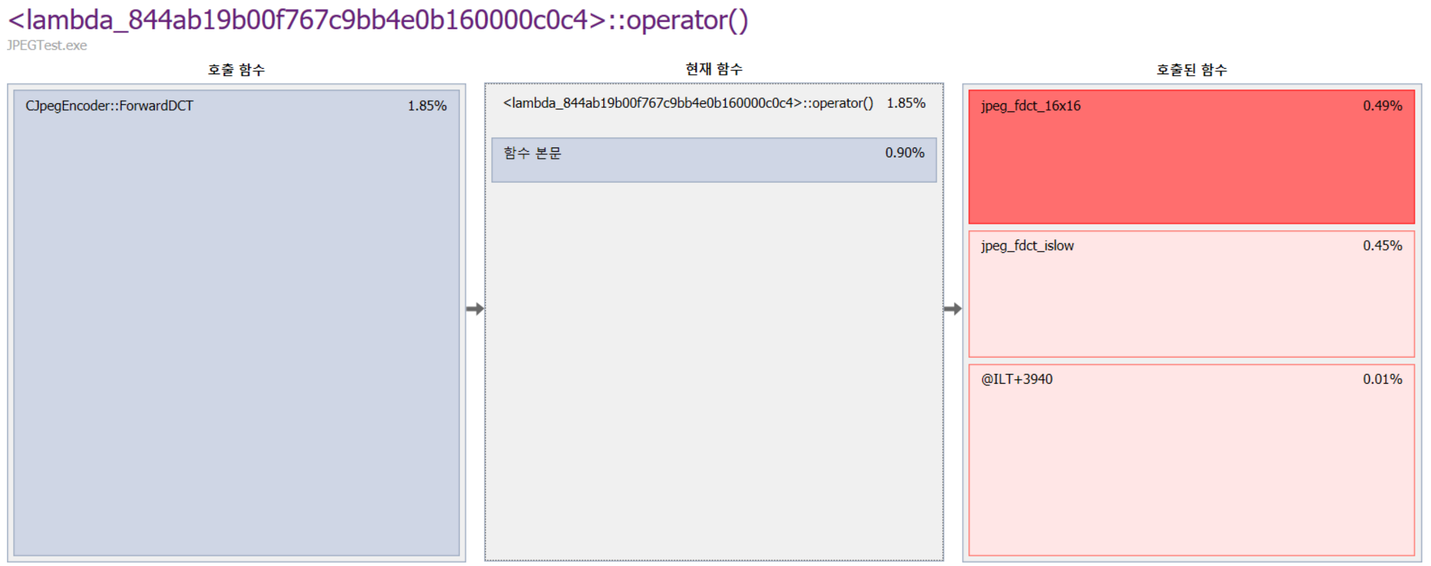
함수 이름	포괄 샘플	전용 샘플	포괄 샘플 비율(%)	전용 샘플 비율(%)
[advapi32.dll]	2007	2007	16.13%	16.13%
[chrome_child.dll]	1639	1408	13.17%	11.31%
[jscript9.dll]	1302	968	10.46%	7.78%
[ntdll.dll]	2591	762	20.82%	6.12%
[win32u.dll]	737	536	5.92%	4.31%
[ntdll.dll]	2802	516	22.51%	4.15%
[d3d10warp.dll]	700	513	5.62%	4.12%
[KernelBase.dll]	380	380	3.05%	3.05%
[mshtml.dll]	1640	280	13.18%	2.25%
[win32u.dll]	903	275	7.26%	2.21%
[chrome_child.dll]	237	193	1.90%	1.55%
[FalloutShelter.exe]	263	147	2.11%	1.18%
[kernel32.dll]	138	138	1.11%	1.11%
[user32.dll]	141	138	1.13%	1.11%
[ntdll.dll]	136	136	1.09%	1.09%
[nvwgf2umx_cfg.dll]	183	115	1.47%	0.92%
<lambda_844ab19b00f767c9bb4e0b160000c0c4...	230	112	1.85%	0.90%
[user32.dll]	133	107	1.07%	0.86%
[chrome.dll]	224	105	1.80%	0.84%
[kernel32.dll]	100	100	0.80%	0.80%
[kernel32.dll]	99	99	0.80%	0.80%
[KernelBase.dll]	87	87	0.70%	0.70%
[msvcp140d.dll]	124	82	1.00%	0.66%
[KernelBase.dll]	81	81	0.65%	0.65%
[msvcrt.dll]	425	80	3.41%	0.64%
[KillerService.exe]	2543	80	20.43%	0.64%
CJpegMCUMatrix::NearestUpscaling	87	78	0.70%	0.63%
CJpegEncoder::rgb_ycc_convert	86	77	0.69%	0.62%
CJpegEncoder::EncodeOneBlock	191	74	1.53%	0.59%
[user32.dll]	73	70	0.59%	0.56%
[node.dll]	68	66	0.55%	0.53%
[KernelBase.dll]	65	65	0.52%	0.52%
[KernelBase.dll]	65	65	0.52%	0.52%
jpeg_fdct_16x16	61	61	0.49%	0.49%
[user32.dll]	68	60	0.55%	0.48%
[user32.dll]	60	56	0.48%	0.45%
jpeg_fdct_islow	56	56	0.45%	0.45%
[WindowsCodecs.dll]	77	55	0.62%	0.44%



성능 프로파일러

현재 뷰: 함수

함수 이름	포괄 샘플	전용 샘플	포괄 샘플 비율(%)	전용 샘플 비율(%)
[advapi32.dll]	2007	2007	16.13%	16.13%
[chrome_child.dll]	1639	1408	13.17%	11.31%
[jscript9.dll]	1302	968	10.46%	7.78%
[ntdll.dll]	2591	762	20.82%	6.12%
[win32u.dll]	737	536	5.92%	4.31%
[ntdll.dll]	2802	516	22.51%	4.15%
[d3d10warp.dll]	700	513	5.62%	4.12%
[KernelBase.dll]	380	380	3.05%	3.05%
[mshtml.dll]	1640	280	13.18%	2.25%
[win32u.dll]	903	275	7.26%	2.21%
[chrome_child.dll]	237	193	1.90%	1.55%
[FalloutShelter.exe]	263	147	2.11%	1.18%
[kernel32.dll]	138	138	1.11%	1.11%
[user32.dll]	141	138	1.13%	1.11%
[ntdll.dll]	136	136	1.09%	1.09%
[nvwgf2umx_cfg.dll]	183	115	1.47%	0.92%
<lambda_844ab19b00f767c9bb4e0b160000c0c4>::operator()	230	112	1.85%	0.90%
[user32.dll]	133	107	1.07%	0.86%
[chrome.dll]	224	105	1.80%	0.84%
[kernel32.dll]	100	100	0.80%	0.80%
[kernel32.dll]	99	99	0.80%	0.80%
[KernelBase.dll]	87	87	0.70%	0.70%
[msvcp140d.dll]	124	82	1.00%	0.66%
[KernelBase.dll]	81	81	0.65%	0.65%
[msvcrt.dll]	425	80	3.41%	0.64%
[KillerService.exe]	2543	80	20.43%	0.64%
CJpegMCUMatrix::NearestUpscaling	87	78	0.70%	0.63%
CJpegEncoder::rgb_ycc_convert	86	77	0.69%	0.62%
CJpegEncoder::EncodeOneBlock	191	74	1.53%	0.59%
[user32.dll]	73	70	0.59%	0.56%
[node.dll]	68	66	0.55%	0.53%
[KernelBase.dll]	65	65	0.52%	0.52%



c:₩work₩dinggul₩jpegtest₩jpegtest₩src₩imageconvert₩jpegencoder.cpp

```
        727     forward_DCT_method_ptr do_dct;
        728     DCTELEM * divisors;
        729
  0.01% 730     auto DoDCT = [&](){
  0.95% 731         ( *do_dct ) ( workspace, pSrc, 0 );
        732
        733         /* Quantize/descale the coefficients, and store into coef_blocks[] */
        734         {
        735             register DCTELEM temp, qval;
        736             register int i;
        737
  0.02% 738             for( i = 0; i < cnDctBlockSize; i++ ) {
  0.07% 739                 qval = divisors[i];
  0.04% 740                 temp = workspace[i];
  0.01% 741                 if( temp < 0 ) {
  0.05% 742                     temp = -temp;
  0.03% 743                     temp += qval >> 1;  /* for rounding */
  0.31% 744                     temp /= qval;
  0.02% 745                     temp = -temp;
        746                 } else {
  0.05% 747                     temp += qval >> 1;  /* for rounding */
  0.22% 748                     temp /= qval;
        749                 }
  0.03% 750                 output_ptr[i] = ( JCOEF ) temp;
  0.05% 751             }
        752         }
        753     };
        754
        755     do_dct = jpeg_fdct_islow;
        756     divisors = &vDCT_YTable[0];
        757     for( int i = 0; i < 2; ++i ){
```

100 %

디스어셈블리 보기

```
                qval = divisors[i];
                temp = workspace[i];
                if( temp < 0 ) {
                    temp = -temp;
                    temp += qval >> 1;  /* for rounding */
                    temp /= qval;
                    temp = -temp;
                } else {
                    temp += qval >> 1;  /* for rounding */
                    temp /= qval;
                }
                output_ptr[i] = ( JCOEF ) temp;
            }
        }
    };

    do_dct = jpeg_fdct_islow;
    divisors = &vDCT_YTable[0];
    for( int i = 0; i < 2; ++i ){
        for( int j = 0; j < 2; ++j ){
            output_ptr = &vMCUBuffer[nBufferIndex++][0];
            pSrc = pDCTSource + 8 * j + i * 128;
            DoDCT();
        }
    }

    do_dct = jpeg_fdct_16x16;
    divisors = &vDCT_CTable[0];
    for( int i = 1; i <= 2; ++i ){
        output_ptr = &vMCUBuffer[nBufferIndex++][0];
        pSrc = pDCTSource + 256 * i;
        DoDCT();
    }
    if( pDCValue != nullptr ){
```

이름	값
do_dct	0xcccccccccccccccc
this	0x00007ff7c0320fd0 {JPEGTest.exe!CJpegEncoder CJpegM

- 빠른 작업 및 리팩터링... Ctrl+.
- 포함 파일의 그래프 생성(I)
- 정의 피킹 Alt+F12
- 정의로 이동(G) F12
- 선언으로 이동(A) Ctrl+F12
- 모든 참조 찾기(A) Shift+F12
- 호출 계층 구조 보기(H) Ctrl+K, Ctrl+T
- 헤더/코드 파일 전환(H) Ctrl+K, Ctrl+O
- 코드 맵
- 중단점(B)
- 다음 문 표시(H) Alt+Num *
- 커서까지 실행(N) Ctrl+F10
- 다음 문 설정(X) Ctrl+Shift+F10
- 디스어셈블리로 이동(D) Alt+G
- 조사식 추가(W)
- 간략한 조사식(Q)... Shift+F9
- 코드 조각(S)
- 잘라내기(T) Ctrl+X
- 복사(Y) Ctrl+C
- 붙여넣기(P) Ctrl+V
- 주석(A)
- 개요(L)
- 다시 검사(R)

```
FF7C02CBC80  mov         dword ptr [nButterIndex],0
egister JCOEFPTR output_ptr{ nullptr };
FF7C02CBC8A  mov         qword ptr [output_ptr],0
nsigned char* pSrc = pDCTSource;
FF7C02CBC95  mov         rax,qword ptr [this]
FF7C02CBC9C  mov         rax,qword ptr [rax+0D8h]
FF7C02CBCA3  mov         qword ptr [pSrc],rax

orward_DCT_method_ptr do_dct;
CTELEM * divisors;

uto DoDCT = [&](){
   ( *do_dct ) ( workspace, pSrc, 0 );

   /* Quantize/descale the coefficients, and store into coef_blocks[] */
   {
       register DCTELEM temp, qval;
       register int i;

       for( i = 0; i < cnDctBlockSize; i++ ) {
           qval = divisors[i];
           temp = workspace[i];
           if( temp < 0 ) {
               temp = -temp;
               temp += qval >> 1;  /* for rounding */
               temp /= qval;
               temp = -temp;
           } else {
               temp += qval >> 1;  /* for rounding */
               temp /= qval;
           }
           output_ptr[i] = ( JCOEF ) temp;
       }
   }
;
FF7C02CBCAA  lea         rax,[output_ptr]
FF7C02CBCB1  mov         qword ptr [rsp+28h],rax
FF7C02CBCB6  lea         rax,[divisors]
FF7C02CBCBD  mov         qword ptr [rsp+20h],rax
FF7C02CBCC2  lea         r9,[pSrc]
FF7C02CBCC9  lea         r8,[workspace]
FF7C02CBCCD  lea         rdx,[do_dct]
FF7C02CBCD4  lea         rcx,[DoDCT]
FF7C02CBCDB  call        <lambda_844ab19b00f767c9bb4e0b160000c0

o_dct = jpeg_fdct_islow;
FF7C02CBCE0  lea         rax,[jpeg_fdct_islow (07FF7C02B35F0h)]
FF7C02CBCE7  mov         qword ptr [do_dct],rax
ivisors = &vDCT_YTable[0];
FF7C02CBCEE  xor         edx,edx
FF7C02CBCF0  lea         rcx,[CJpegEncoder::vDCT_YTable (07FF7C
FF7C02CBCF7  call        std::array<int,64>::operator[] (07FF7C
FF7C02CBCFC  mov         qword ptr [divisors],rax
or( int i = 0; i < 2; ++i ){
FF7C02CBD03  mov         dword ptr [rbp+204h],0
or( int i = 0; i < 2; ++i ){
FF7C02CBD0D  jmp         CJpegEncoder::ForwardDCT+0DDh (07FF7C0
FF7C02CBD0F  mov         eax,dword ptr [rbp+204h]
FF7C02CBD15  inc         eax
FF7C02CBD17  mov         dword ptr [rbp+204h],eax
FF7C02CBD1D  cmp         dword ptr [rbp+204h],2
FF7C02CBD24  jge         CJpegEncoder::ForwardDCT+19Eh (07FF7C02CBDDEh)
```

- 소스 코드로 이동(G)
- 간략한 조사식(Q)... Shift+F9
- 중단점(B)
- 다음 문 표시(H) Alt+Num *
- 커서까지 실행(N) Ctrl+F10
- 플래그 지정된 스레드를 커서까지 실행(F)
- 다음 문 설정(X) Ctrl+Shift+F10
- ✓ 주소 표시(A)
- ✓ 소스 코드 표시(C)
- 코드 바이트 표시(Y)
- ✓ 기호 이름 표시(S)
- 줄 번호 표시(L)
- ✓ 도구 모음 표시(T)

최적화에 관계된 것들

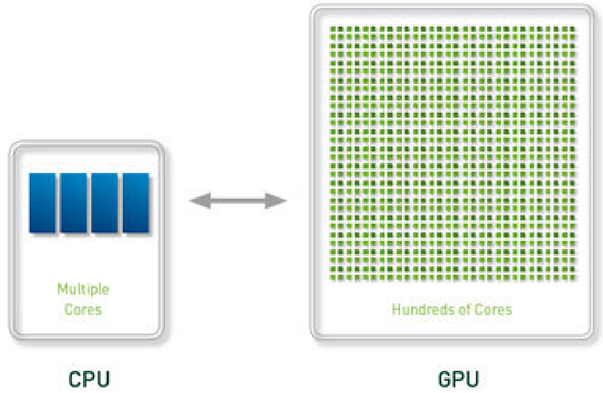
1. 프로세서
2. 저장장치
3. OS
4. 알고리즘
5. 프로그래머

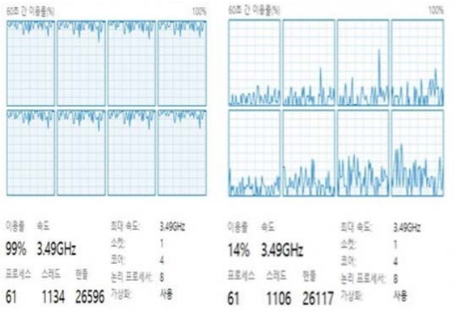
최적화에 관계된 하드웨어 특성들

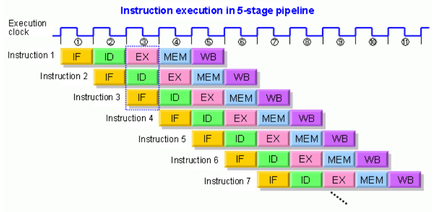
1. 프로세서
2. 저장장치
3. OS

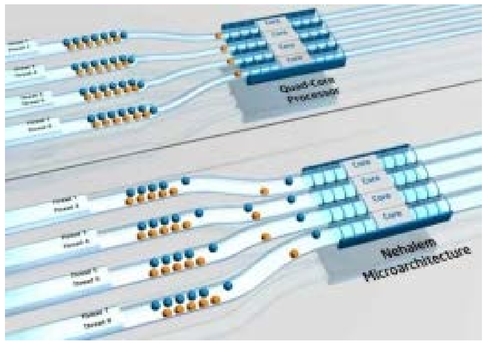
프로세서

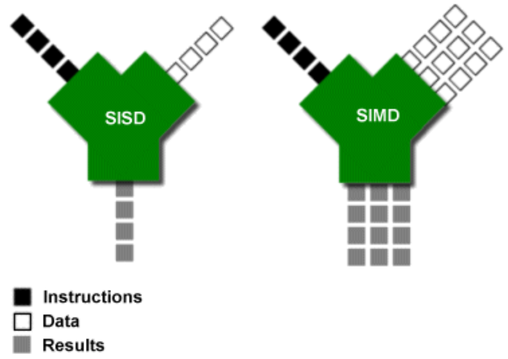
Multi Core
Instruction Pipe-line
Simultaneous Multithreading (SMT)
SIMD
GPGPU



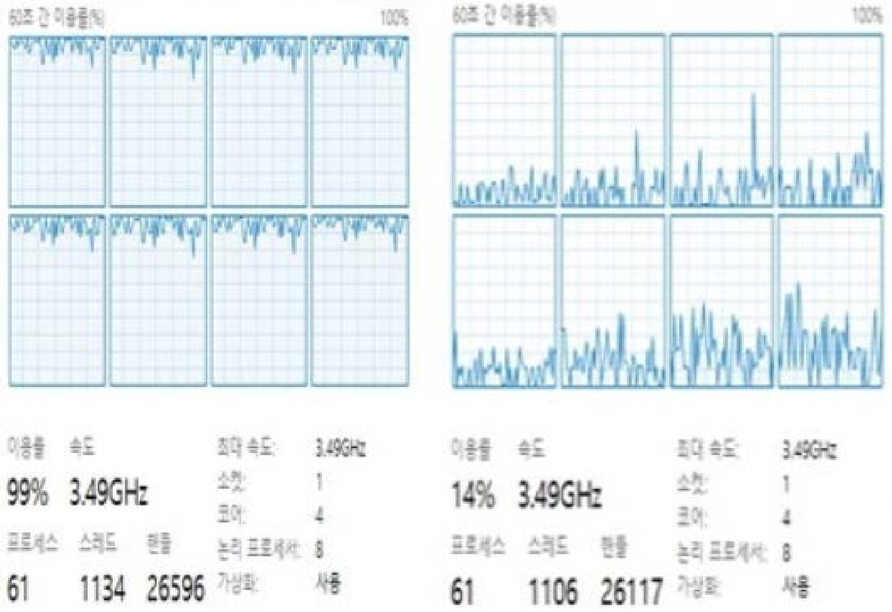






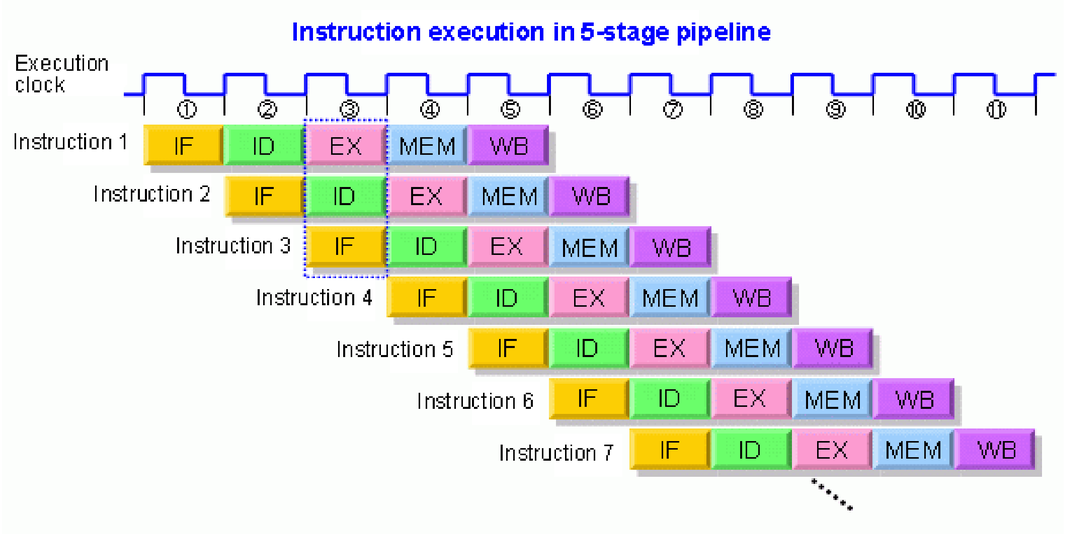


Multi-core processor



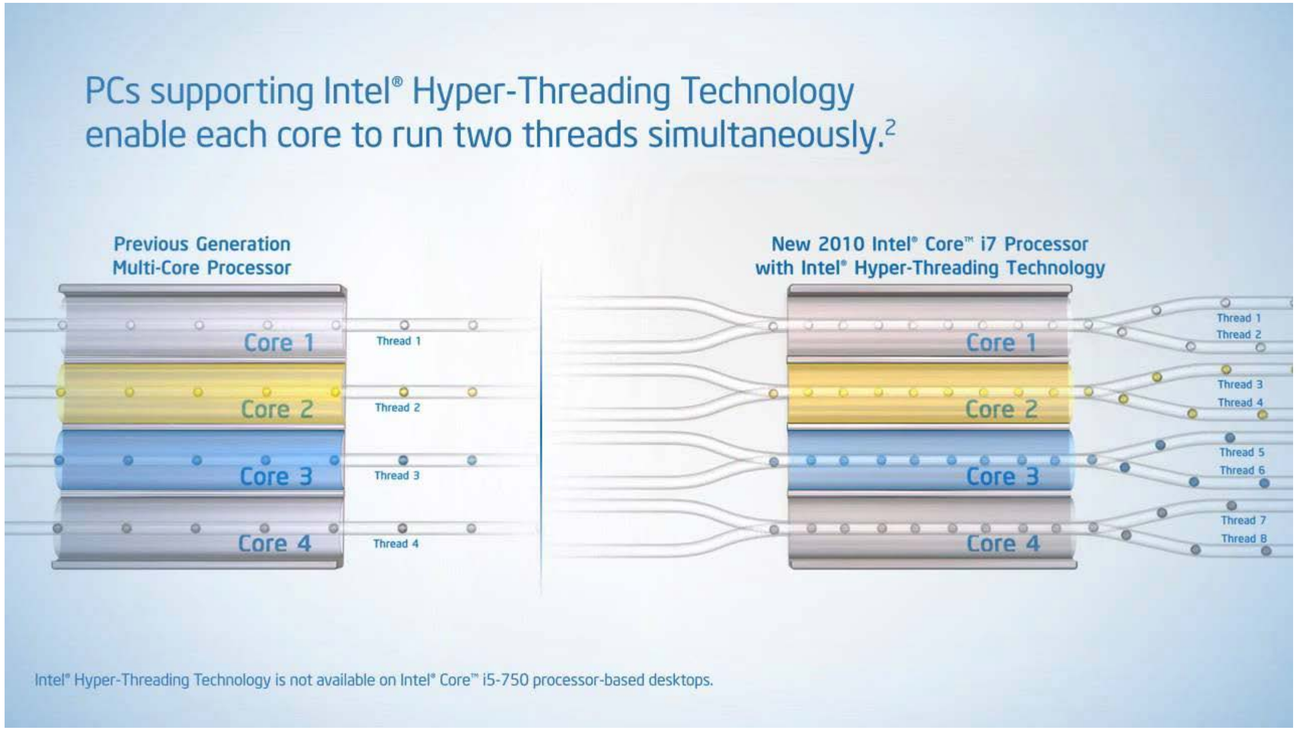
Instruction Pipe-line

Instruction Fetch
Instruction Decode
Execution
Memory Access
Write Back



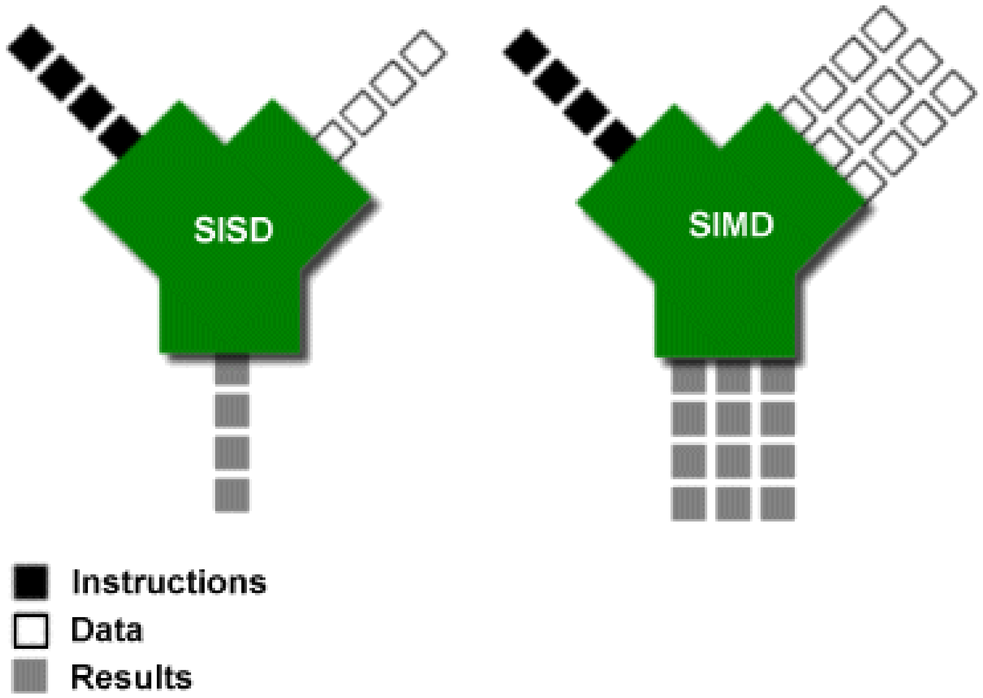
Simultaneous Multithreading (SMT)

Hyper-threading



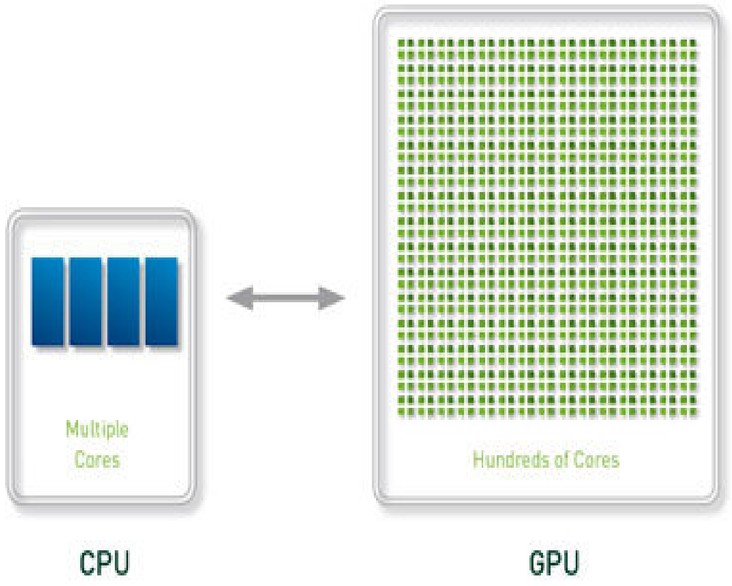
Single Instruction Multiple Data (SIMD)

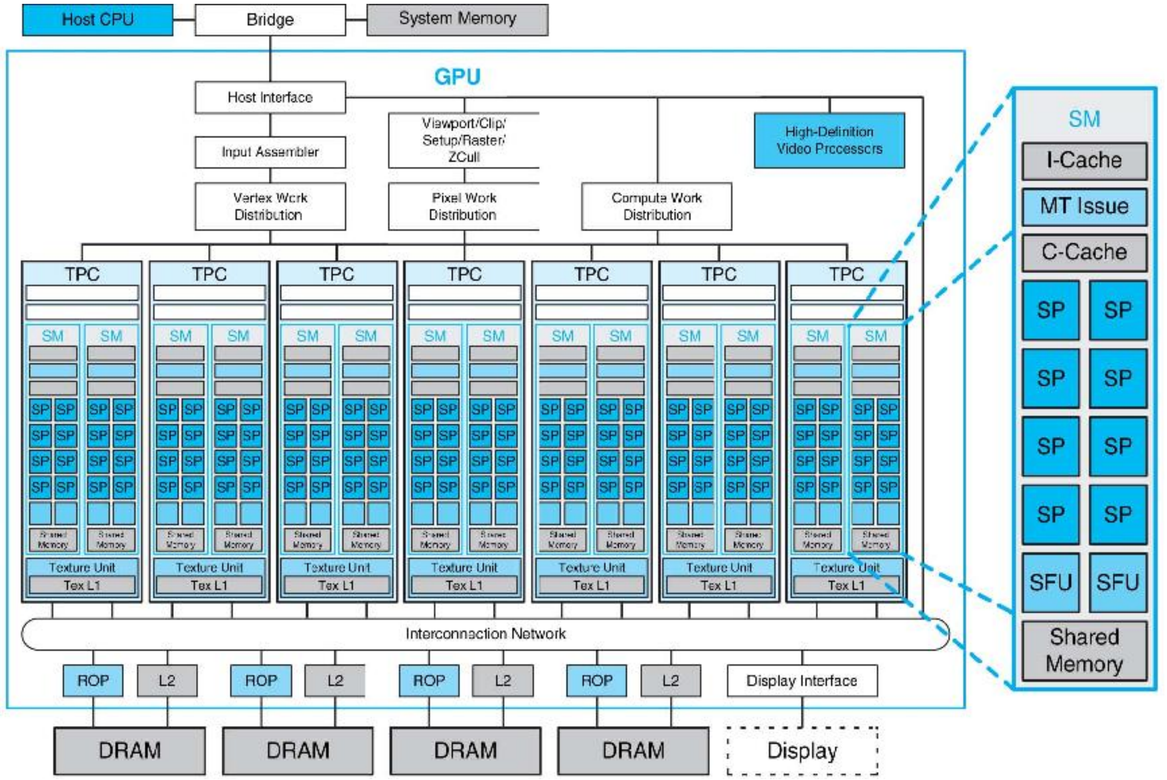
MMX



General-Purpose computing on Graphics Processing Units (GPGPU)

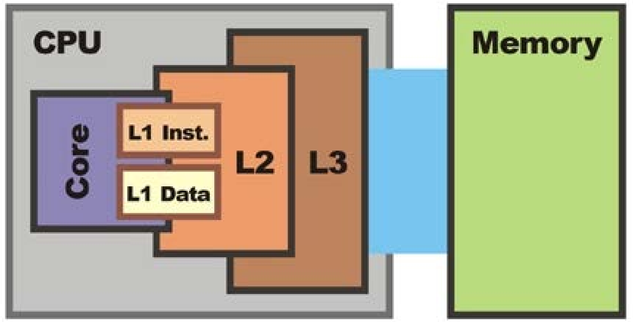
CUDA

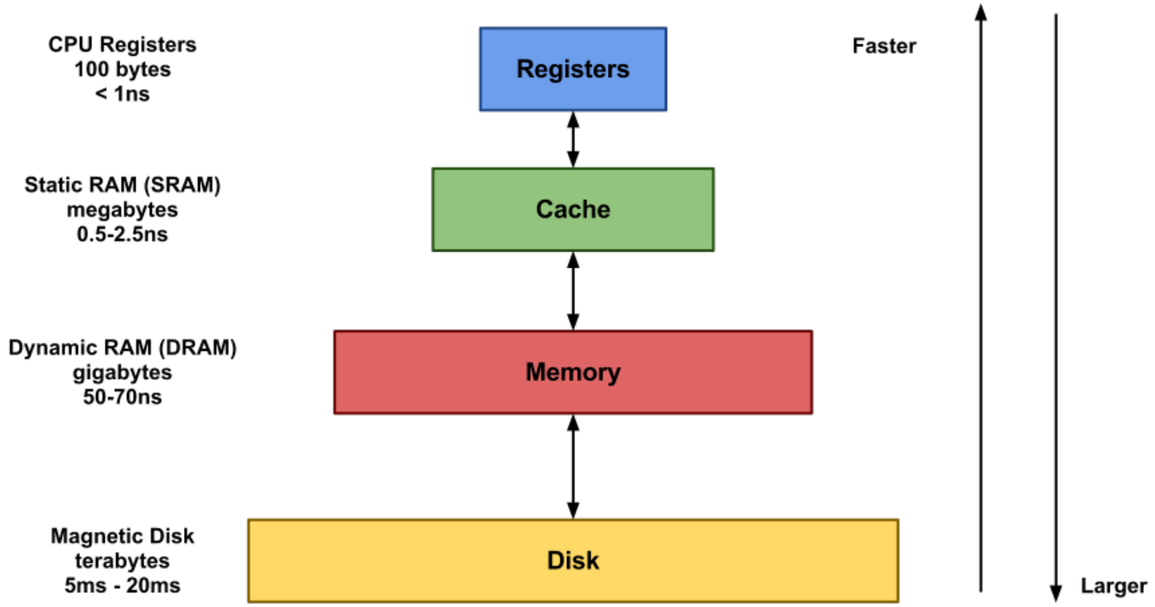




저장장치

Cache
Memory BUS
File System





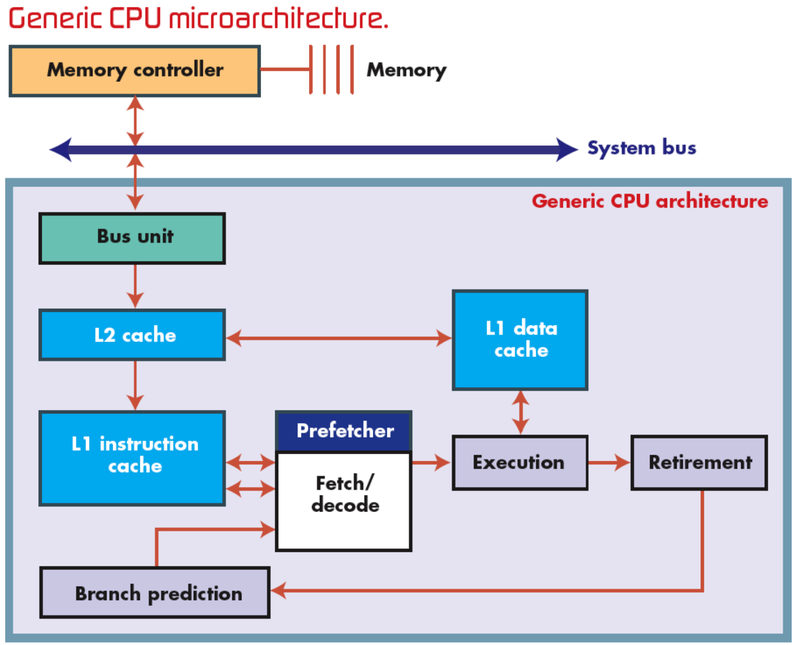
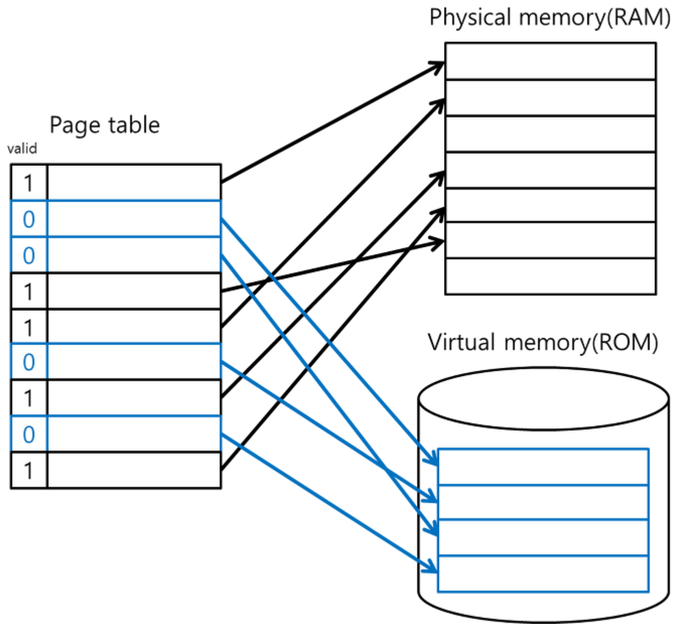
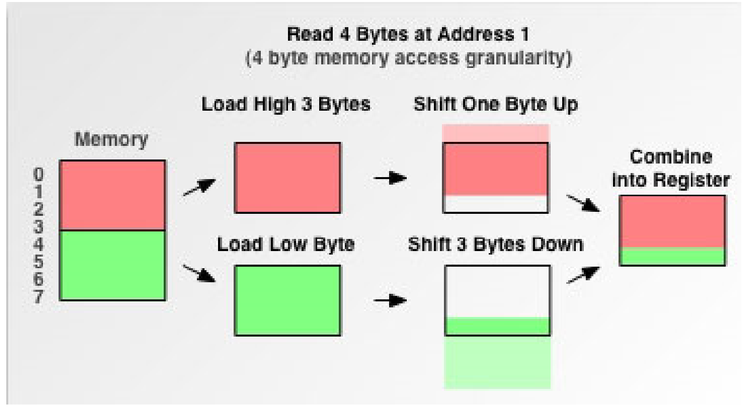


Figure 1

OS

Virtual Memory
Word Alignment
Allocation
Memory Copy





C++와의 연관

1. 프로세서
2. 저장장치
3. OS

프로세서

Instruction Fetch
Instruction Decode
Execution
Memory Access
Write Back

Thread
Lambda

저장장치

Cache
Memory BUS
File System

STL
Constructor/destructor

OS

Virtual Memory
Word Alignment
Allocation
Memory Copy

STL
Constructor/destructor

Optimize C++

1. Memoization & Lazy evaluation
2. Thread 활용
3. STL
4. Construction / Destruction / new / delete
5. Deep Copy / Reference / move / forward
6. Lambda
7. Pointer indexing
8. Dynamic binding / dynamic casting

기본기

Memoization

동일한 계산을 반복해야 할 때, 이전에 계산한 값을 메모리에 저장함으로써 동일한 계산의 반복 수행을 제거하여 프로그램 실행 속도를 빠르게 하는 기술

Lazy evaluation

계산의 결과값이 필요할 때까지 계산을 늦추는 기법

Thread 활용 <>

STL <>

Object Life-cycle

<>

Copy/Move <>

Dynamic binding / Dynamic casting

<>

vector <>

감사합니다.