CHUNGNAM NATIONAL UNIVERSITY

시스템 프로그래밍

강의 1. 과목 소개

http://eslab.cnu.ac.kr

오리지날@CMU, 2015, Randy Bryant

- https://scs.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=d8c83d3a-8074-4afe-ae3b-693e2250999a

이 과목의 주제

- 컴퓨터 프로그래밍은 추상화(Abstraction)이 대부분
  - 그러나, 본질을 놓치면 안된다
- 컴퓨터 전공의 추상화
  - 자료구조
  - 가상메모리 ?
  - 아이폰의 앱들 – 카카오톡, 페이스북
- 추상화는 한계가 있다
  - 버그가 있을 때
  - 시스템 내부의 동작에 대한 이해가 필요할 때
- 과목의 목표
  - 컴퓨터하드웨어와 소프트웨어에 대한 본질을 이해
  - 프로그램이 컴퓨터에서 실행된다는 것에 대한 실체적 진실에 접근
  - 효율적인 프로그래머를 만들자
    - 기괴한 버그를 효과적으로 퇴치
    - 내가 작성한 프로그램이 무슨 일을 하고 있는지에 대한 심오한 이해제공
    - 성능과 자원의 한계를 고려한 프로그래밍
    - 컴퓨터에서 일어나고 있는 현상에 대한 총체적 이해의 근거를 제시

비전공자는 이해하기 어려운 사실 #1

- *정수가 정수가 아니고, 실수가 실수가 아니다?*
- **Examples**
  - **$x^2 \geq 0$?**
    - Float's: Yes!
    - Int's:
      - 40000 * 40000 --> 1600000000
      - 50000 * 50000 --> ??
  - **(x + y) + z = x + (y + z)?**
    - Unsigned & Signed Int's: Yes!
    - Float's:
      - (1e20 + -1e20) + 3.14 --> 3.14
      - 1e20 + (-1e20 + 3.14) --> ??

컴퓨터 내에서의 연산

- 수학에서는 의미 없는 이상한 값이 갑자기 생기지는 않는다
  - 수식연산은 중요한 수학법칙을 갖는다
- 그러나 컴퓨터에서는 일반적인 수학 법칙을 가정할 수 없다
  - 데이터 표현의 유한성 때문에
- 관찰(Observation)
  - 어떤 추상화가 어떤 상황에 적용된 것인지 이해하는 것이 필요
  - 컴파일러 개발자나 고급 프로그래머들에게는 중요한 주제

비전공자는 이해하기 어려운 사실 #2

- *어셈블리어를 알아야 한다 !!!*
- 대개의 경우, 어셈블리 프로그램을 작성할 가능성은 ..... **zero !**
  - 컴파일러가 훨씬 더 우수하고 참을성이 있다
- 어셈블리어를 이해하는 것은 하드웨어 수준의 실행모델을 이해하는데 매우 중요
  - 버그가 있을 때 프로그램의 동작
    - 고차원 언어 모델로는 이해할 수 없다
  - 프로그램의 성능을 튜닝할 때
    - 프로그램의 효율성을 이해하기 위해
  - 시스템 소프트웨어 구현시
    - 컴파일러는 기계어 코드가 목적코드이다
    - 운영체제는 프로세스의 상태를 관리해야 한다
  - **malware를 만들거나 malware와 싸우기 위해**
    - x86-64 어셈블리를 배운다

어셈블리 코드 예제

```
;*************************************************************************
;Initialize SCCs
;       SCC channels 0, 1, 2, and 3 are initialized. It is assumed
;       that the Peripheral chip selects have been programmed to begin
;       at 0x8000, and the SCCs are PCS0 and PCS1.
;       SPEED: 9600 bits/second from x16 BRG clock from 7.372800 PCLCK
;       CHARS: 8 bits. No parity. 1 Stop bit.
;
;       No interrupts are used.
;*************************************************************************
init_c0:        mov     dx,CH0_CTL              ;address of scc to init
                mov     cx,offset init_c1       ;'return' address
                jmp     init_scc
init_c1:
                mov     dx,CH1_CTL              ;address of scc to init
                mov     cx,offset init_c2       ;'return' address
                jmp     init_scc

init_c2:        mov     dx,CH2_CTL              ;address of scc to init
                mov     cx,offset init_c3
                jmp     init_scc

init_c3:        mov     dx,CH3_CTL
                mov     cx,offset init_muxs
                jmp     init_scc
```

비전공자는 이해하기 어려운 사실 #3

- *메모리가 중요하다*

- 메모리크기는 무한하지 않다
  - 메모리는 할당해주고 관리해야 한다
  - 많은 응용프로그램들은 메모리에 영향 받는다
- 메모리 참조 버그는 치명적이다
  - 버그의 결과가 시공간적으로 연관성이 없다
- 메모리 성능은 일정하지 않다
  - 캐시와 가상메모리 효과는 성능에 비선형적 영향을 미친다
  - 프로그램들을 메모리 시스템의 특성에 최적화 시키면 성능이 대폭 개선된다

메모리 참조 버그

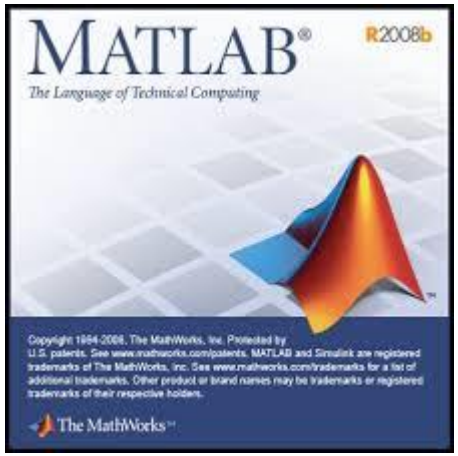
```
double fun(int i)
{
  volatile double d[1] = {3.14};
  volatile long int a[2];
  /* Possibly out of bounds */
  a[i] = 1073741824; return d[0];
}
```

```
fun(0)   ->    3.14
fun(1)   ->    3.14
fun(2)   ->    3.1399998664856
fun(3)   ->    2.0000061035156
fun(4)   ->    3.14, then segmentation fault
```

2015 ESC Silicon Valley

@2015 ESC Silicon Valley

MathWorks®



메모리 시스템 성능

```
void copyij(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (i = 0; i < 2048; i++)
    for (j = 0; j < 2048; j++)
      dst[i][j] = src[i][j];
}
```

```
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (j = 0; j < 2048; j++)
    for (i = 0; i < 2048; i++)
      dst[i][j] = src[i][j];
}
```

59,393,288 clock cycles

1,277,877,876 clock cycles

(Measured on 2GHz Intel Pentium 4)

21.5 **배 더 느리다**!

- 계층적 메모리 구조를 이해하는 사람과 그렇지 못한 사람
- 성능은 메모리 접근 방식에 영향을 받는다
  - 다중 배열에서 어떻게 인덱싱 하는가에 따라 달라진다

비전공자는 이해하기 어려운 사실 #4

- *컴퓨터는 프로그램 실행 이외의 여러가지 일을 한다*

- 컴퓨터는 데이터의 입출력이 필요하다
  - I/O 시스템은 프로그램의 신뢰성과 성능에 매우 중요하다
- 프로그램이 순차적으로 실행되는 것은 아니다
  - 예외적인 순서로(돌발!) 진행될 수 있어야 컴퓨터 다워진다

이 과목의 목표

- 프로그램이 어떻게 하드웨어에서 동작하는지 이해한다.
  - 어셈블리 프로그래밍을 이용해서
- 리눅스의 프로그래밍 환경에 익숙해 지도록 한다
  - 리눅스에서 프로그래밍할 가능성이 높다
- 컴퓨터 시스템을 활용하는 프로그래밍 기법을 학습한다
  - 프로세스의 제어 **Process control**
  - 시그널 **Signal**
  - 메모리 관리 **Memory management**
- 컴퓨터 하드웨어와 소프트웨어의 본질을 이해한다

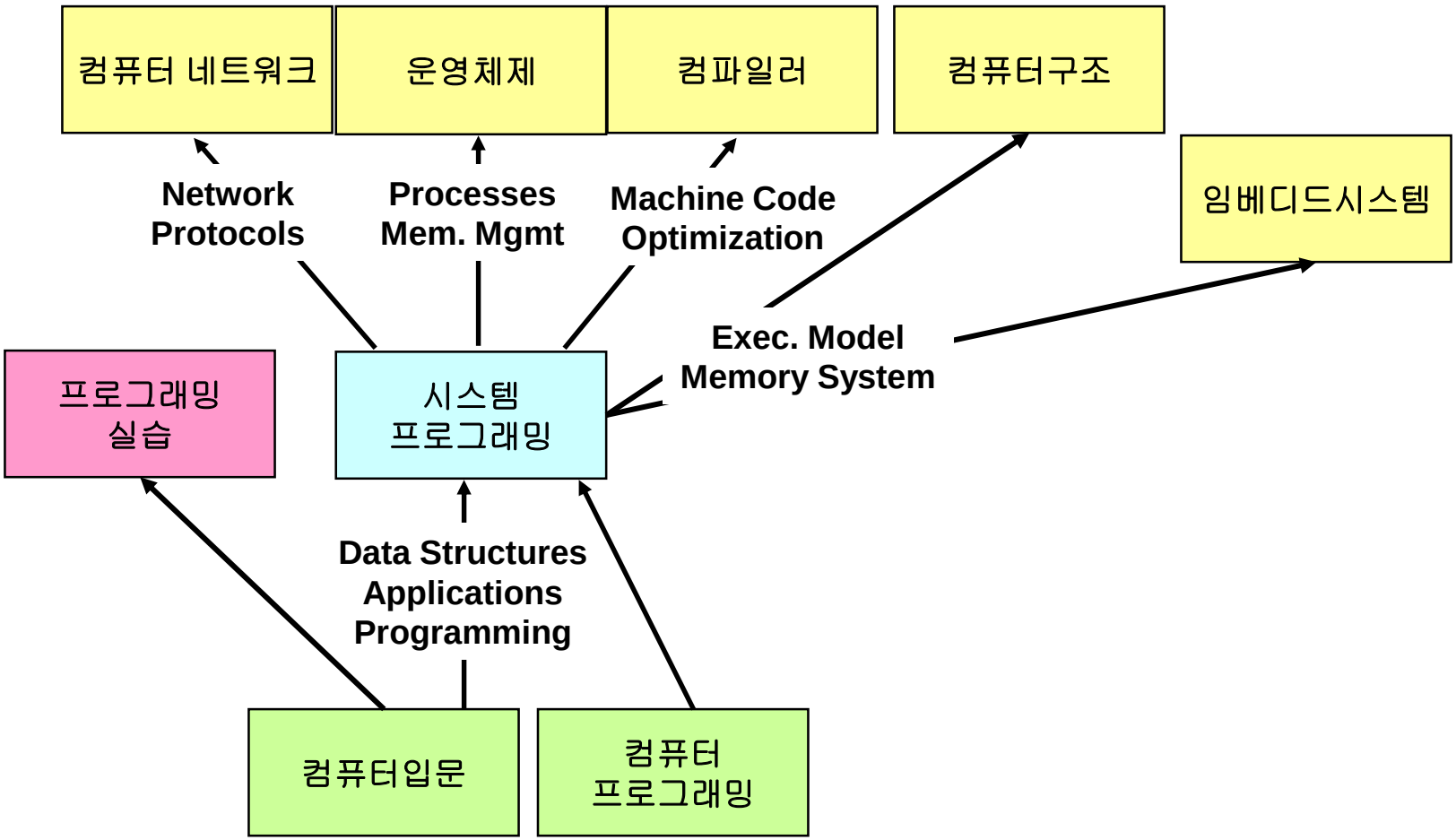
컴퓨터 시스템에 관한 깊은 이해를 제공한다

이 수업이 끝날 때 여러분은,

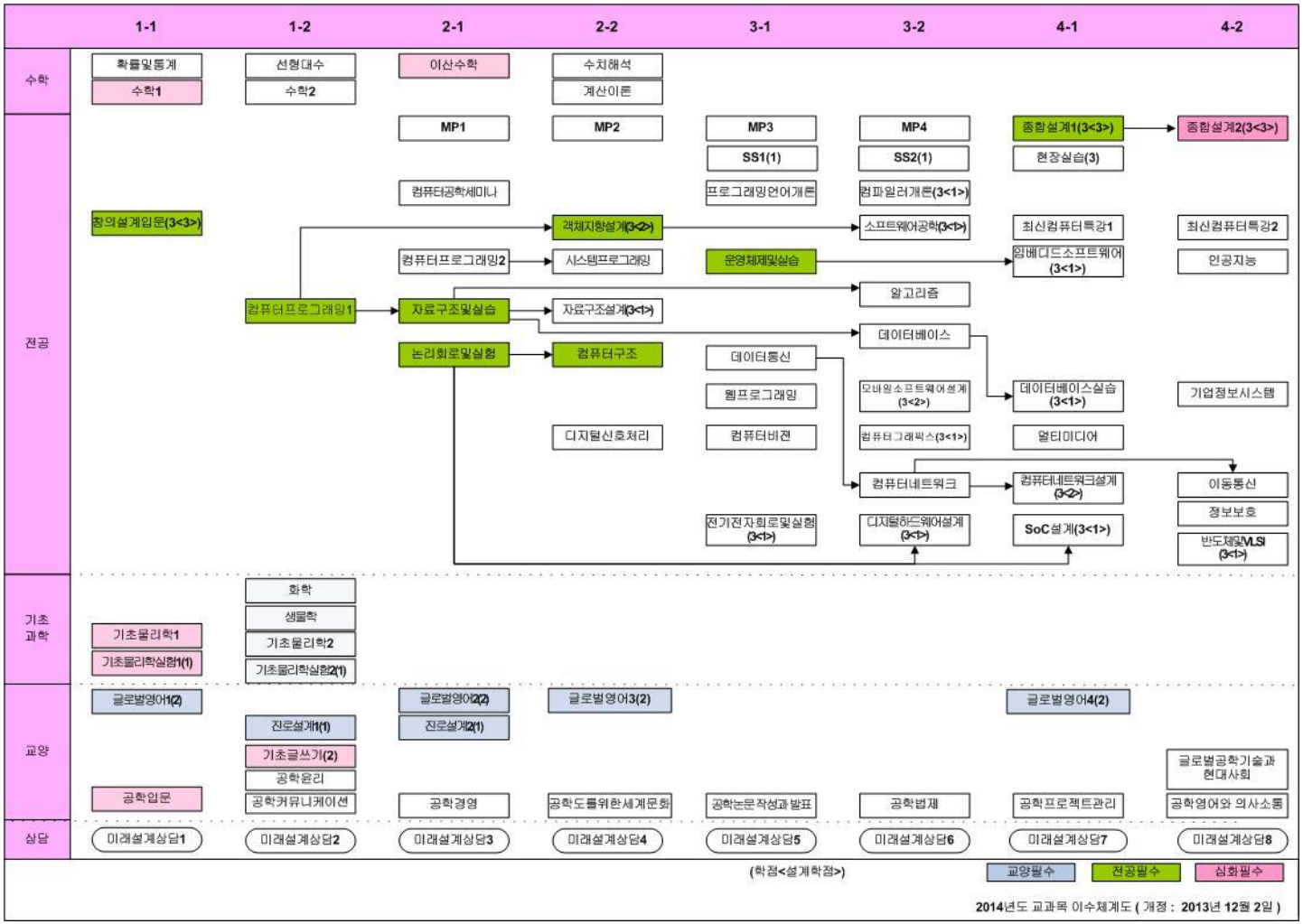
- 리눅스에서 C 프로그래밍 하는 기술을 습득한다
- x86-64 어셈블리어를 사용할 수 있게 된다.
- 다른 프로세서의 어셈블리어를 쉽게 습득할 수 있게 된다.
- 고급 디버깅 기술을 사용할 수 있게 된다.
- 컴퓨터 내부에서의 숫자의 표현방법을 이해하게 된다
- 고급언어가 어떻게 번역되고, 컴퓨터에서 처리되는지 이해하게 된다.
- 운영체제의 예외처리 과정을 이해한다
- 프로세스를 제어하는 프로그램을 작성할 수 있다
- 운영체제의 메모리 관리 원리를 이해하게 된다

막강한 준-프로 시스템 프로그래머가 된다!

컴퓨터 전공과목과의 결정적 관계



교과목 이수체계도(2014학번)



강의 일정

주	날짜	강의실 (화)	날짜	실습실 (목)
1	9월 5일	1.Intro		
2	9월 12일	2.정수	9월 10일	리눅스 개발환경 익히기
3	9월 19일	3.부동소숫점	9월 17일	GCC & Make
4	9월 26일	추석휴일	9월 24일	추석휴일
5	10월 3일	개천절	10월 1일	(이론)4. 어셈1 – move
6	10월 10일	5.어셈2 – 제어문	10월 8일	데이타랩
7	10월 17일	6.어셈3 – 프로시져 I	10월 15일	어셈블리어/GDB
8	10월 24일	7.어셈3 – 프로시져 II	10월 22일	폭탄랩
9	10월 31일	중간고사(저녁 7시)	10월 29일	
10	11월 7일	8.프로세스 1	11월 5일	
11	11월 14일	9.프로세스 2	11월 12일	Tiny shell 1
12	11월 21일	10.시그널	11월 19일	Tiny shell 2
13	11월 28일	11.동적메모리 1	11월 26일	Tiny shell 3
14	12월 5일	12.동적메모리 2	12월 3일	Malloc lab1
15	12월 12일	기말고사(저녁7시)	12월 17일	Malloc lab2
16	12월 19일	Wrap-up/종강	12월 13일	Malloc lab3

담당교수

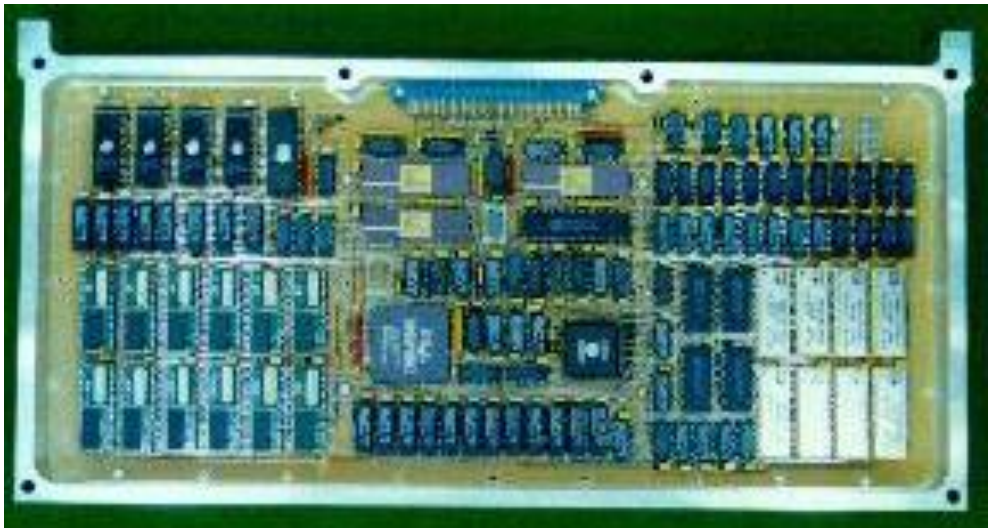
Hyungshin Kim, Professor of CSE, CNU

BS	Computer Science at KAIST
MS	Satellite Communication Engineering, Univ. of Surrey, UK
PhD	Computer Science at KAIST
90-92	Researcher, SSC, UK
92-01	Researcher, SaTReC, KAIST
03-04	Post-Doc, Carnegie Mellon University, USA
08-09	Visiting Researcher, Rutgers University, USA

Office : E5 521
Voice : 821-5446
E-mail : hyungshin@cnu.ac.kr
Web : http://eslab.cnu.ac.kr
Lab : Embedded System Lab. (ESL), E5 533
Office hour : TBD

아리안과 나

나와 우주



National Space Lab
NSL

과목 개요 – 1/2

- 조교 : 하동휘(01반), 장혁수*(00반, 2018 1학기 우수조교)
- 튜터 : 김성민(2017 No.2), 이수정(2017 No.2)

- 교재 :
  - Computer systems : A programmer's perspective, 3판, by Randal E. Bryant and David O'Hallaron, Prentice Hall
- 참고문헌
  - Computer systems : A programmer's perspective, 2판, by Randal E. Bryant and David O'Hallaron, Prentice Hall

- 교과목 홈페이지
  - 충남대학교 사이버캠퍼스
  - http://e-learn.cnu.ac.kr/lms/class/classroom/

- 선수지식
  - 능수 능란한 C 언어 프로그래밍 능력

과목 개요 – 2/2

- 성적평가
  - 중간고사 : 30%
  - 기말고사 : 30%
  - 실습 : 30%
    - 4 Major Labs 중 2개 이상 미제출은 F : DataLab, BombLab, ShellLab, MallocLab
  - 출석 : 10%
  - 주의 !
    - 2학년이 아닌 고학년생들은 별도의 그룹으로 상대평가 하게 됨
    - 재이수학생들은 별도 평가하며, A는 특별한 경우에만 부여. 90%는 B+이하 부여
- 출석
  - 2회까지는 눈 감아줌. 3회 결석부터 2점씩 감점
  - 실습포함 8회이상 결석시 F(학칙에 의거)
- 숙제
  - 예습 교재읽기와 실습숙제
  - 마감일을 넘기는 경우 50% 감점, 그 이후부터는 1일에 10% 씩 감점
- 숙제/실습 복사에 대해
  - 부도덕한 행위에 대해서 F 부과(실습 숙제 카피 주의!!!)
  - 주요 실습 평가시 구두 평가 실시 – 구두 평가 결과로 실습의 진정성 평가

Syste

감투상	한정원	정진휘	
우수상	김성민	이수정	
최우수상	손예지	이예은	

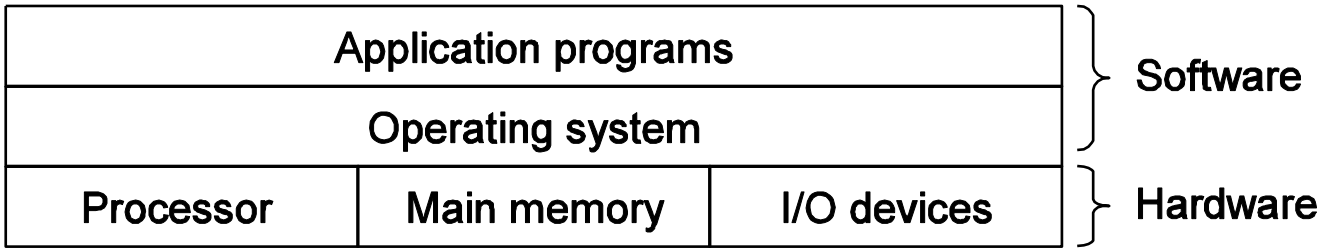
2018 시스템 프로그래밍 강의 구성

- 25분 강의 3분 휴식, 20분강의 (분위기 봐가면서)
- 난이도 높은 실습에 대한 지도 강화 => 실습 풀이시간
- 실습의 카피 여부 집중 검증 => **GIT** 사용 의무화
  - 쉘랩 부터 사용 개시
- 사이버 캠퍼스의 **QandA** 사용
- 2학년 학생들을 배려하는 성적평가
- 어리버리한 2학년들의 전공으로의 연착륙

제 1장. 컴퓨터 시스템 개관

- 컴퓨터 시스템의 본질을 찾아서...

컴퓨터 시스템의 계층도


Application programs
Operating system
Processor
Main memory
I/O devices
Software
Hardware

컴퓨터 하드웨어의 구조

- 하드웨어 구성요소
  - 버스 **buses**
  - 입출력 장치, **I/O devices**
  - 주기억장치, **main memory**
  - 캐쉬, **cache memory**
  - 중앙처리장치, **CPU (Central Processing Unit)**
- 버스
  - 구성요소간의 정보교환 통로
  - 종류 : 입출력버스, 시스템 버스

프로세서

- 중앙처리장치 Central Processing Unit : CPU
  - 산술논리연산장치 : arithmetic and logic unit, ALU
  - 제어장치 : Control unit
  - 레지스터
    - 프로그램 카운터 PC
    - 상태 레지스터
    - 메모리 주소 레지스터
    - 메모리 데이터 레지스터
    - 명령 레지스터, instruction pointer, IP
    - 범용 레지스터, general-purpose register

CPU

- 주요기능
  - 기억장치로부터 명령/데이터 호출 및 저장
  - 명령의 해석 : 제어장치
  - 명령의 실행 : **ALU**
  - 입출력 연산 : **I/O**
- 명령 주기 **Instruction cycle**
  - 인출 **fetch** : **PC**에 의거하여 명령어를 가져옴
  - 해독 **decode** : 명령어를 해석하여 제어 신호 생성
  - 실행 **execution** : 가져온 명령어를 실행
  - 결과저장 **store** : 실행 결과를 저장

운영체제 Operating System

- 운영체제란?
  - 응용프로그램이 하드웨어를 효율적으로 사용할 수 있도록 도와주는 프로그램
- 목적
  - 하드웨어 추상화 **Hardware Abstraction**
  - 시스템 인터페이스 추상화
- 운영체제의 관리대상
  - 프로세스
  - 메모리
  - 입출력장치
  - 소프트웨어 자원
    - 파일시스템, 라이브러리, 유틸리티

Hello world ?!!

- 위대한 프로그램 **hello.c**

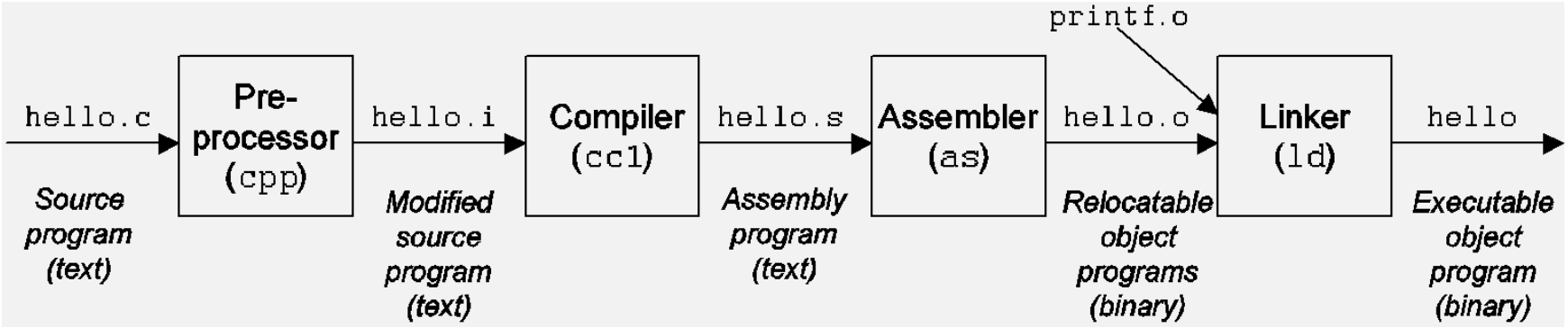
```
#include <stdio.h>

int main()
{
        printf("hello, world□n");
}
```

- 본질 ! 컴퓨터의 모든 정보는 비트, 바이트로 저장된다

프로그램은 변신한다

- hello.c 컴파일 과정
  - unix> gcc –o hello hello.c



- 본질 ! 컴파일러의 뒷면에는 이렇게 복잡한 일이 일어나고 있다

어셈블리 프로그램 ? 실행 목적 프로그램 ? 기계어 프로그램 ?

컴파일 과정을 이해하는 것은 매우 중요하다

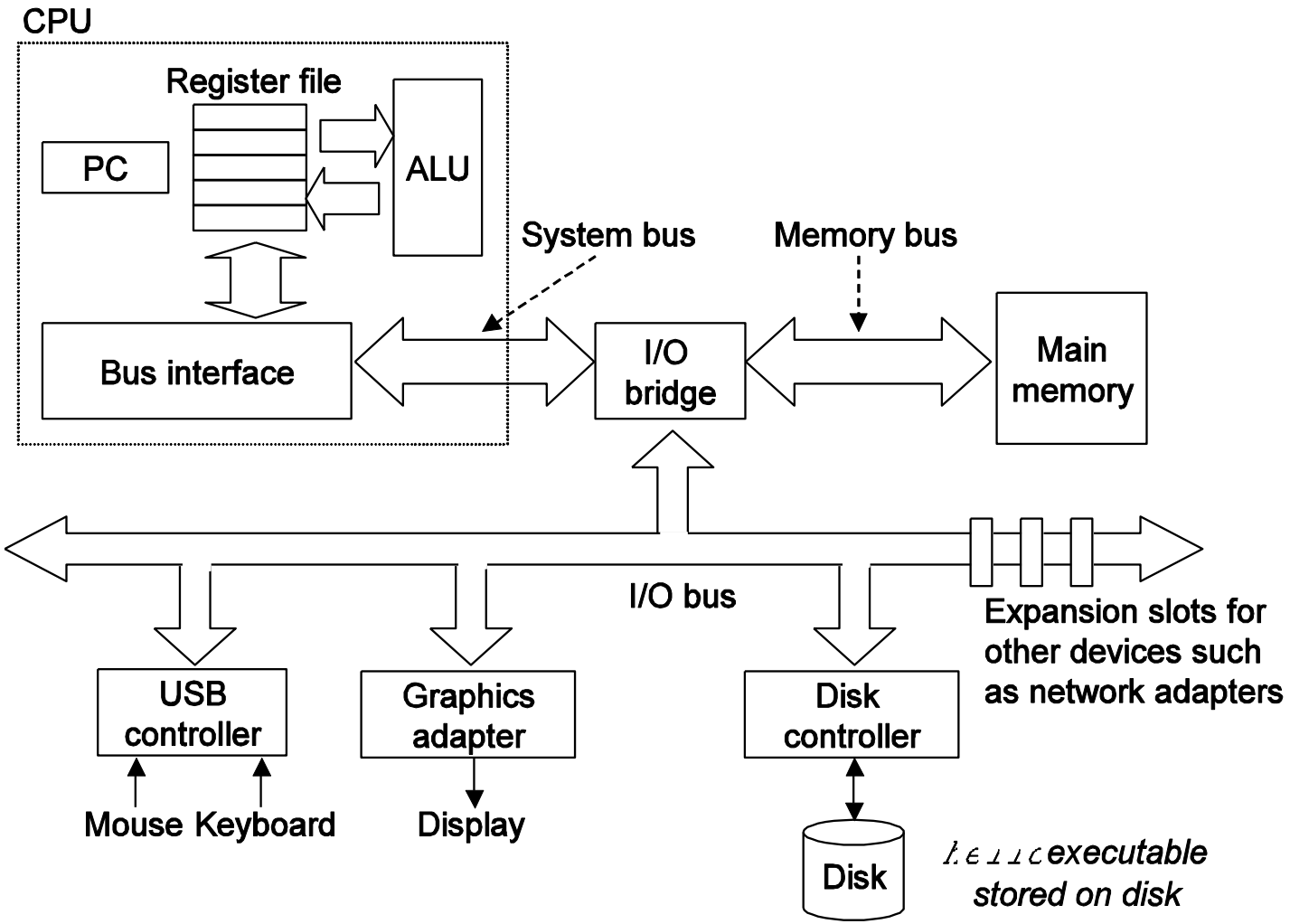
- 프로그램이 컴퓨터 내부에서 어떻게 처리되는지 이해할 수 있다
- 대규모 프로그램 개발 과정을 이해할 수 있다
- 프로그램 성능을 개선할 수 있다
- 링크시에 발생하는 에러를 이해할 수 있다

변환된 프로그램의 실행

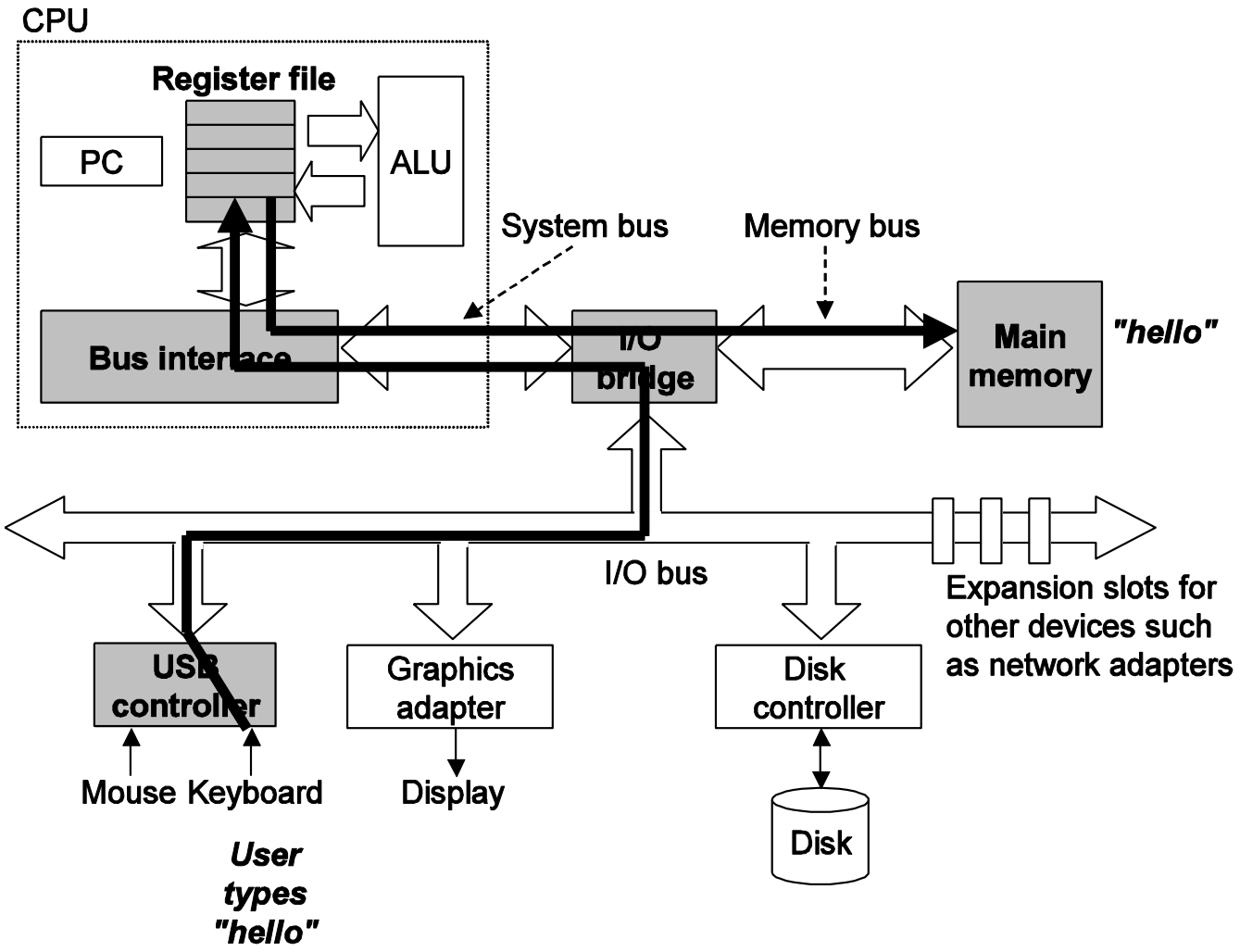
```
cesl-linux> ./hello
hello, world
cesl-linux>
```

- 리눅스 쉘에서 실행파일 이름을 치면 실행됨
  - 쉘이 여러분이 입력한 파일 이름을 찾아서, 그것이 실행파일이면 프로그램을 로드하여 실행함
  - **hello** 프로그램이 실행되면서 결과가 화면에 출력됨.
  - 쉘이 다음 명령을 기다리고 있음
- 본질?

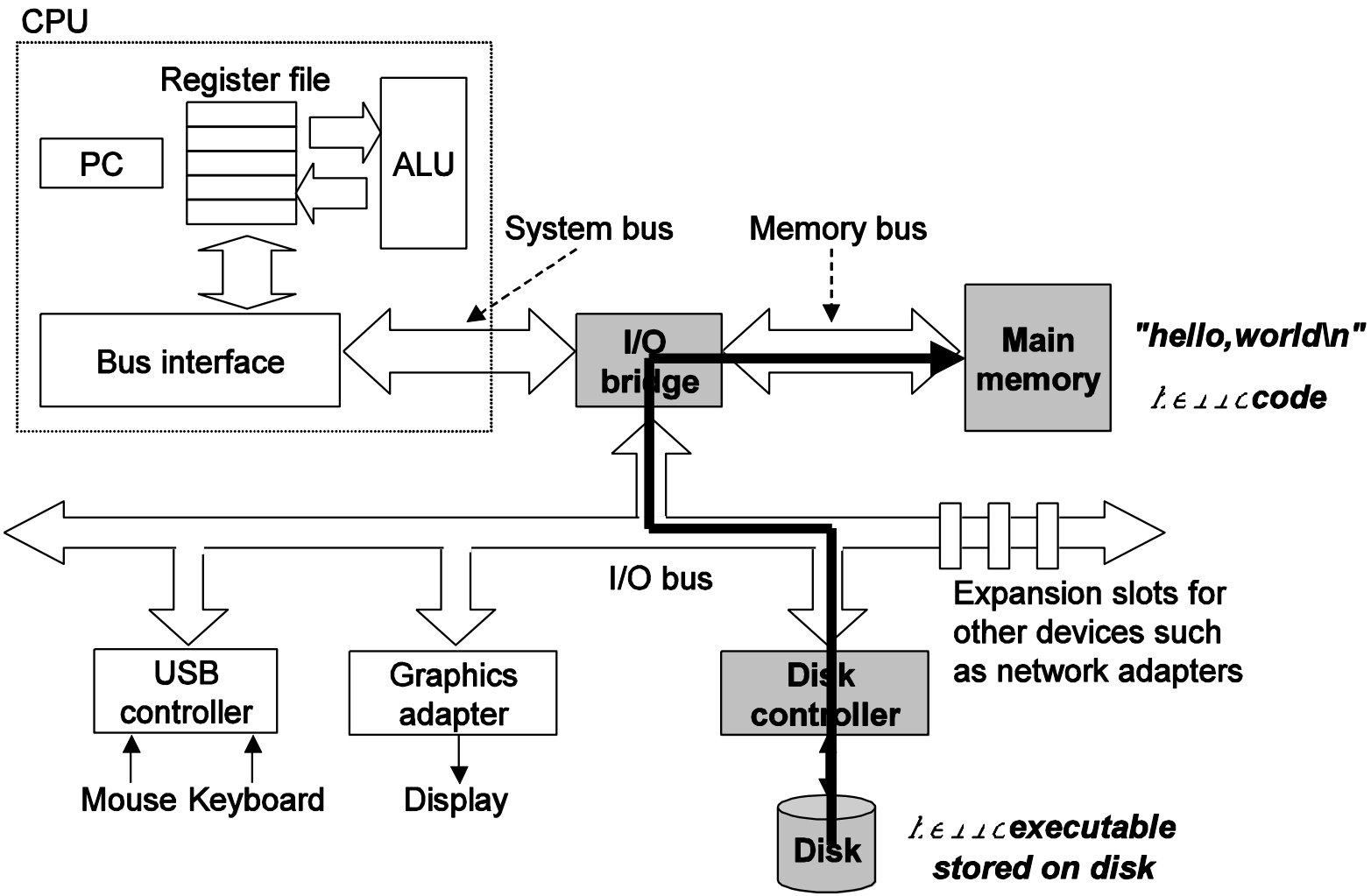
컴퓨터의 구조


CPU
Register file
PC
ALU
System bus
Memory bus
Bus interface
I/O bridge
Main memory
I/O bus
Expansion slots for other devices such as network adapters
USB controller
Graphics adapter
Disk controller
Mouse Keyboard
Display
Disk
hello executable stored on disk

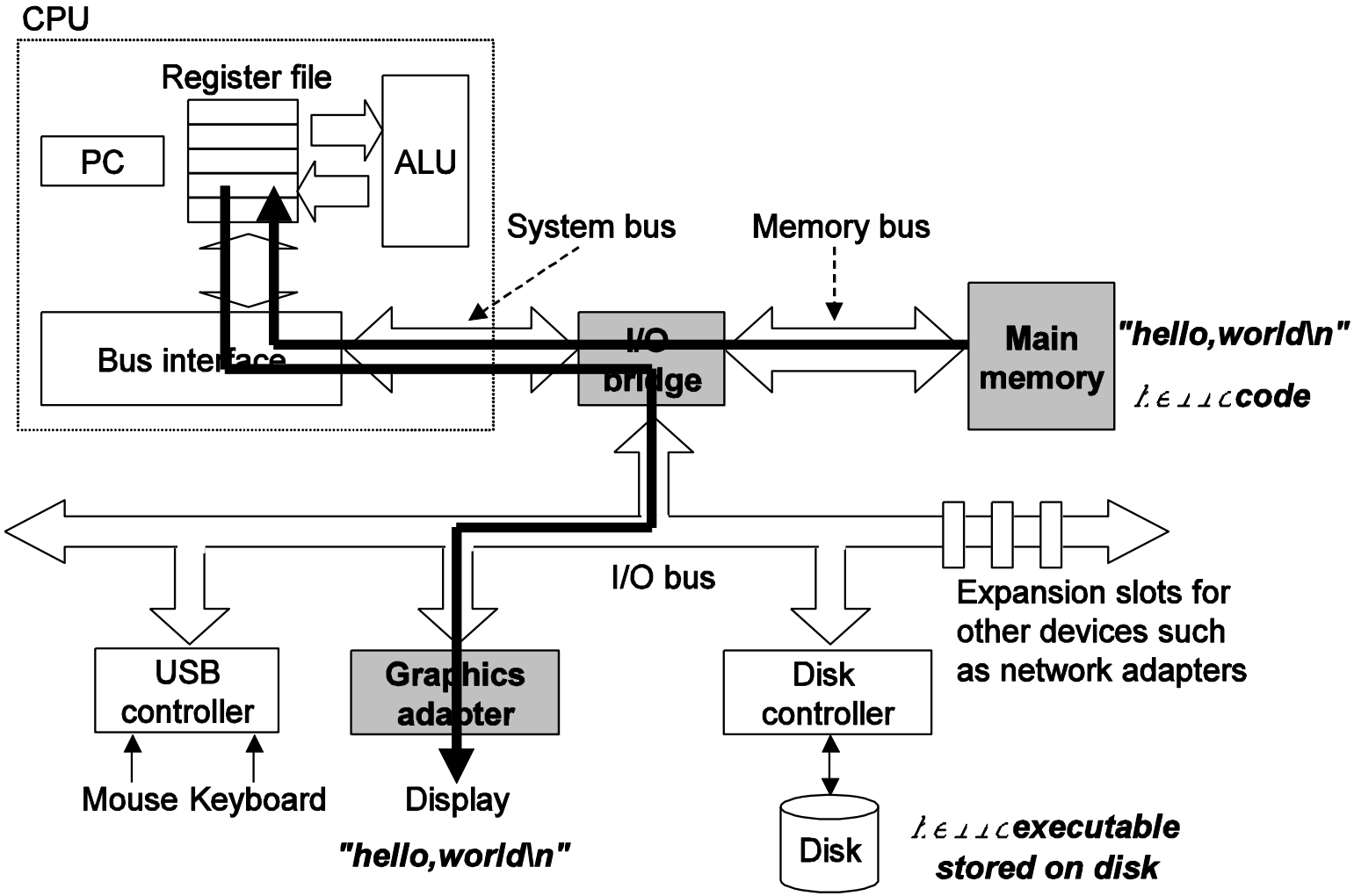
실행의 본질 : hello 명령의 인식



실행의 본질 : hello 프로그램의 로딩



실행의 본질 : hello 프로그램의 실행



요약

- 컴퓨터 시스템은 하드웨어, 시스템 소프트웨어, 응용프로그램으로 구성된다.
- 프로그램은 번역되어 변화한다.
- 프로그램은 시스템 프로그램의 도움으로 하드웨어에서 실행된다.

다음주를 위한 준비

- 새 학기를 맞는 여러분의 결심
  - 강의는 열심히, 생활은 즐겁게
  - 전공으로 진입하는 중요한 기회의 학기
  - 적절한 수준의 수강신청
- 사이버 캠퍼스 홈페이지 가서 강의자료 내려받기.

C 프로그래밍 능력 진단 퀴즈

- 아래의 call_swap( ) 함수는 zip1과 zip2에 저장된 정수값을 포인터를 이용해서 위치를 바꿔서 저장해주는 함수 swap( )를 호출한다. Swap( ) 함수를 구현하시오. Swap 함수 호출후에 zip2=15213, zip1=91125가 저장되어야 한다.

```
int zip1 = 15213;
int zip2 = 91125;

void call_swap()
{
  swap(&zip1, &zip2);
}
```

```
void swap(                    )
{




}
```

C 프로그래밍 능력 진단 퀴즈 2

- 키보드로 문자열을 입력을 받아, 입력받는 문자를 화면에 찍어주는 다음의 함수를 작성하시오. 단, 이 함수는 이 입력작업을 키보드에서 **CTRL-C**가 들어올 때까지 무한 반복해야 한다.

```
void getInput(  )
{




}
```

C 프로그래밍 능력 진단 퀴즈 3

- 100명의 시험점수가 저장되어 있는 배열 score[100] 를 매개변수로 넘겨받아, 이 배열내의 데이터 중에서 최고점수와 최소점수를 찾아서 화면에 출력하는 MinMax( ) 함수를 아래 함수를 이용해서 작성하시오.

```
void MinMax(int *score)
{




}
```