操作系统研讨课
Course: B62011Y

蒋德钧

Fall Term 2017-2018

email: jiangdejun@ict.ac.cn
office phone: 62601007

中国科学院大学
University of Chinese Academy of Sciences

Lecture 0 Introduction

2017.09.13

中国科学院大学
University of Chinese Academy of Sciences

Lecture 0: Introduction

- Overview
  - Course introduction
  - Course administration

Lecture 0: Introduction

- Why?
- What?
- How?

Lecture 0: Introduction

- Why do I need to learn this course?
  - Getting credits
  - Capabilities of system developing
  - Basis of full stack view

Lecture 0: Introduction

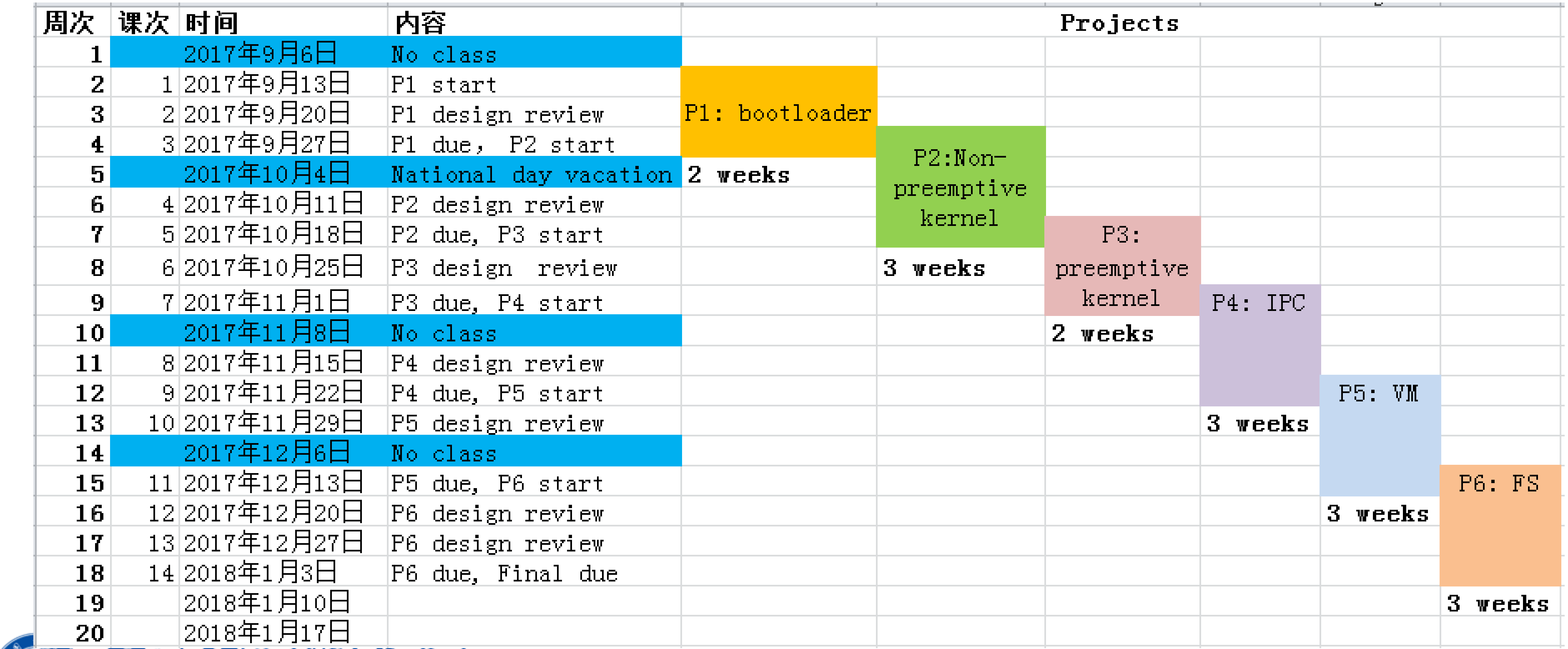
- What do I learn in this course?
  - How to build a simple operating system
    - Bootloader
    - Kernel supporting multiprogramming
    - Process communication and management
    - Virtual memory management
    - File system

Lecture 0: Introduction

- How do I finish this course?
  - Think and Practice
  - Group working

Lecture 0: Introduction

- Course administration
  - Classrooms：教 205（机房） & 221（机房）
  - Schedule



周次	课次	时间	内容	Projects					
1		2017年9月6日	No class						
2	1	2017年9月13日	P1 start	P1: bootloader					
3	2	2017年9月20日	P1 design review						
4	3	2017年9月27日	P1 due, P2 start		P2:Non-preemptive kernel				
5		2017年10月4日	National day vacation	2 weeks					
6	4	2017年10月11日	P2 design review						
7	5	2017年10月18日	P2 due, P3 start			P3: preemptive kernel			
8	6	2017年10月25日	P3 design review		3 weeks				
9	7	2017年11月1日	P3 due, P4 start				P4: IPC		
10		2017年11月8日	No class			2 weeks			
11	8	2017年11月15日	P4 design review						
12	9	2017年11月22日	P4 due, P5 start					P5: VM	
13	10	2017年11月29日	P5 design review				3 weeks		
14		2017年12月6日	No class						
15	11	2017年12月13日	P5 due, P6 start						P6: FS
16	12	2017年12月20日	P6 design review					3 weeks	
17	13	2017年12月27日	P6 design review						
18	14	2018年1月3日	P6 due, Final due						
19		2018年1月10日							3 weeks
20		2018年1月17日							

Lecture 0: Introduction

- Course administration
  - Lecturer
    - 蒋德钧：jiangdejun@ict.ac.cn
  - Teaching assistant
    - Dr. 朱晓静 (assistant professor)：zhuxiaoj@ict.ac.cn
    - 王盈(Ph.D student)：wangying01@ict.ac.cn
    - 潘海洋(Ph.D student)：panhaiyang@ict.ac.cn
    - 李天祥(Ph.D student)：litianxiang@ict.ac.cn
  - Office hour
    - Make appointment

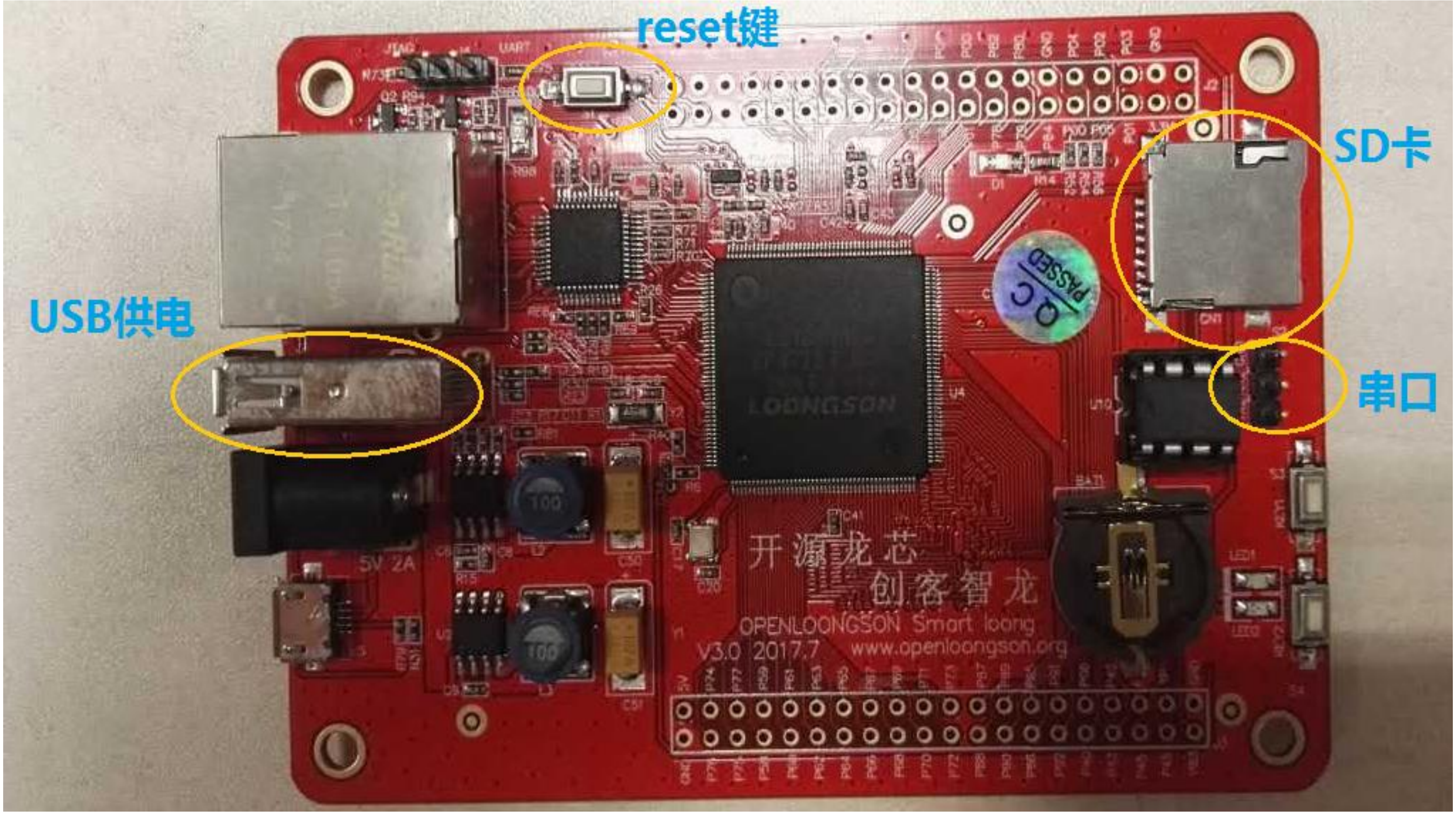
Lecture 0: Introduction

- Development environment
  - Software
    - Linux operating system with PC in the lab or your own laptop
    - Virtual machine with VirtualBox or Physical machine
    - Ubuntu 16.04

Lecture 0: Introduction

- Development environment
  - Hardware
    - One piece of Openloongson SoC board
    - One USB cable
    - One serial port cable
    - One SSD card and one card reader
    - Protection package

Lecture 0: Introduction



Lecture 0: Introduction

- Grouping
  - P1 ~ P2: grouping I
  - P3 ~ P4: grouping II
  - P5 ~ P6: grouping III
  - Group students randomly
  - Group presentation + individual submission

Lecture 0: Introduction

- Project submission
  - Design documents
  - Source code + README
  - Submission site: course web site
    - http://sep.ucas.ac.cn/

Lecture 0: Introduction

- Grading
  - Grading per project
    - design review: 40 points
    - code development: 60 points
  - Final grading
    - Basic * 0.9 + Bonus * 0.1
    - Basic
      - P1: 10%
      - P2 ~ P4: 15%
      - P5: 20%
      - P6: 25%
    - Bonus: depends on projects

Lecture 0: Introduction

- Grading
  - Grading individually depends on
    - group presentation and Q&A
    - project submission
  - Submit your project on time: 100%
  - Submit one week after deadline: -30%
  - Copying others' code is ABSOLUTELY prohibited
    - No points will be given

Lecture 0: Introduction

- Daily Q&A
  - WeChat
  - QQ

操作系统研讨课2017秋

该二维码7天内(9月20日前)有效，重新进入将更新

Lecture 0: Introduction

- Any question?

Lecture 1 Bootloader

2017.09.13

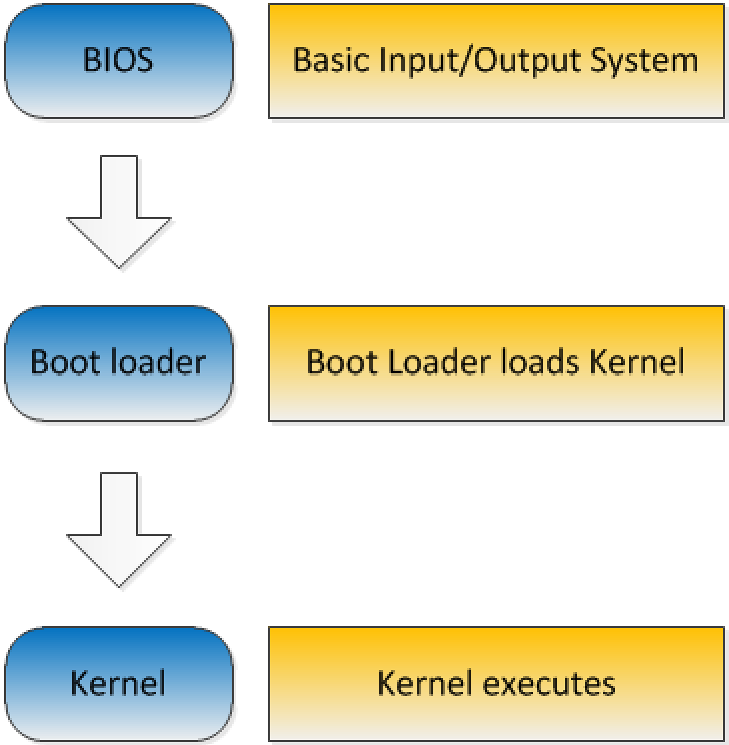
中国科学院大学
University of Chinese Academy of Sciences

Project 1 – Bootloader

- Requirements
  - Write a bootloader to start a simple kernel based on Openloongson SoC board
    - bootblock.s
    - createimage.c

Project 1 – Bootloader

- Booting procedure



Project 1 – Bootloader

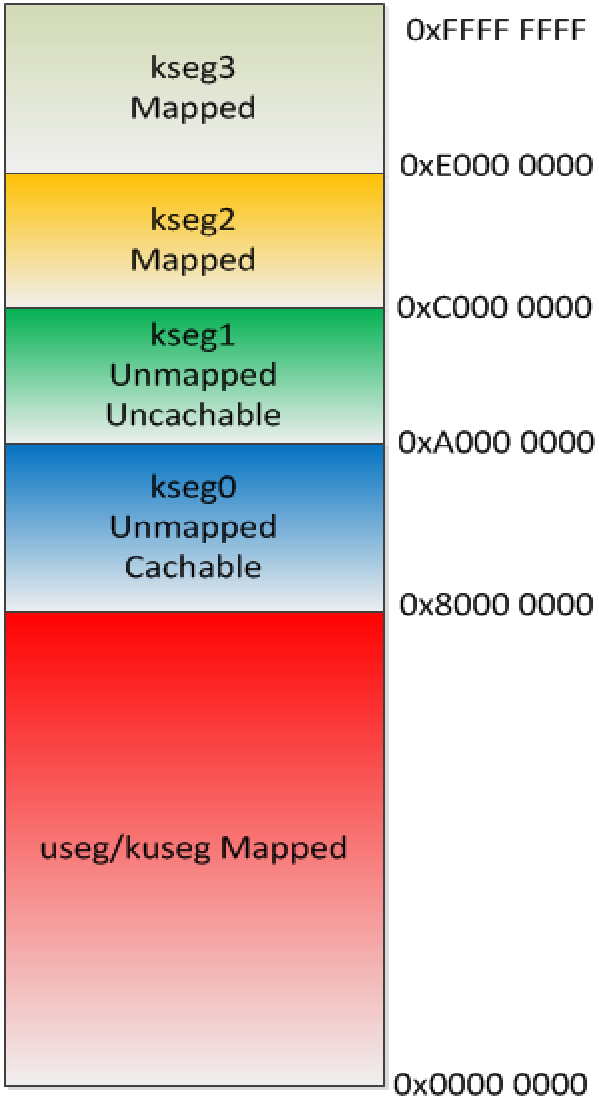
- BIOS
  - Basic Input/Output System
  - Firmware used to perform hardware initialization after power-on
  - Load bootloader
- Bootblock
  - Loaded by BIOS
  - Hard disk

Project 1 – Bootloader

- Bootloader
  - A small program to enable operating system
    - Load the kernel
    - Set up a stack space
    - Switch control to the kernel

Project 1 – Bootloader

- Memory mapping



kseg3 Mapped

kseg2 Mapped

kseg1 Unmapped Uncachable

kseg0 Unmapped Cachable

useg/kuseg Mapped

0xFFFF FFFF

0xE000 0000

0xC000 0000

0xA000 0000

0x8000 0000

0x0000 0000

Project 1 – Bootloader

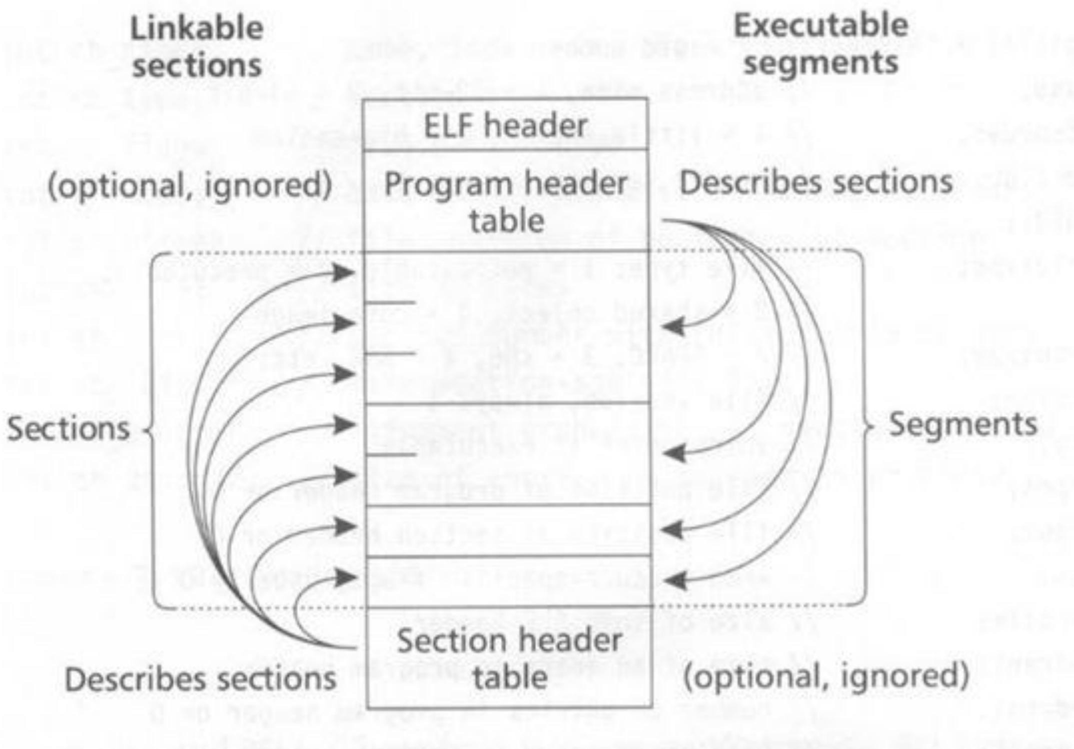
- Createimage
  - Executable file
    - gcc
  - Bootable OS image
    - createimage tool

Project 1 – Bootloader

- ELF object file format
  - Executable and Linking Format (ELF)
  - Object file
    - Binary representation of programs
  - Created by assembler and link editor

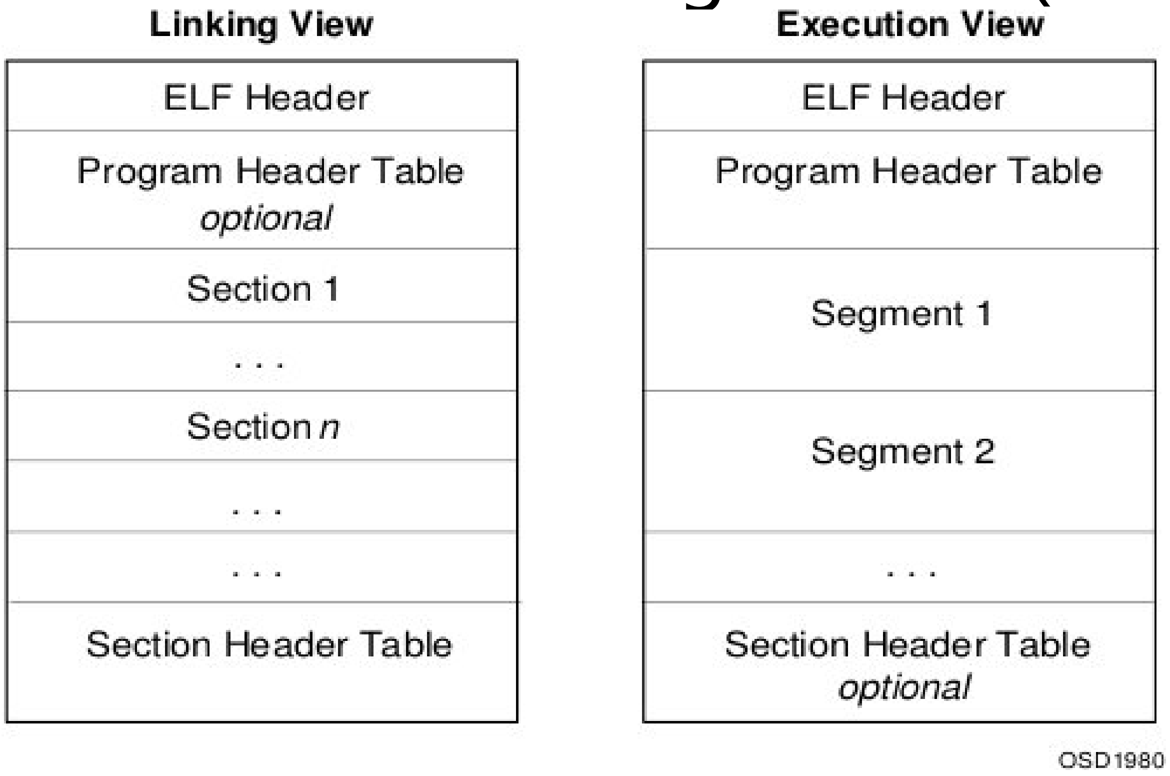
Project 1 – Bootloader

- ELF object file format
  - Executable and Linking Format (ELF)



Project 1 – Bootloader

- ELF object file format
  - Executable and Linking Format (ELF)



Linking View

ELF Header

Program Header Table
optional

Section 1

. . .

Section *n*

. . .

. . .

Section Header Table

Execution View

ELF Header

Program Header Table

Segment 1

Segment 2

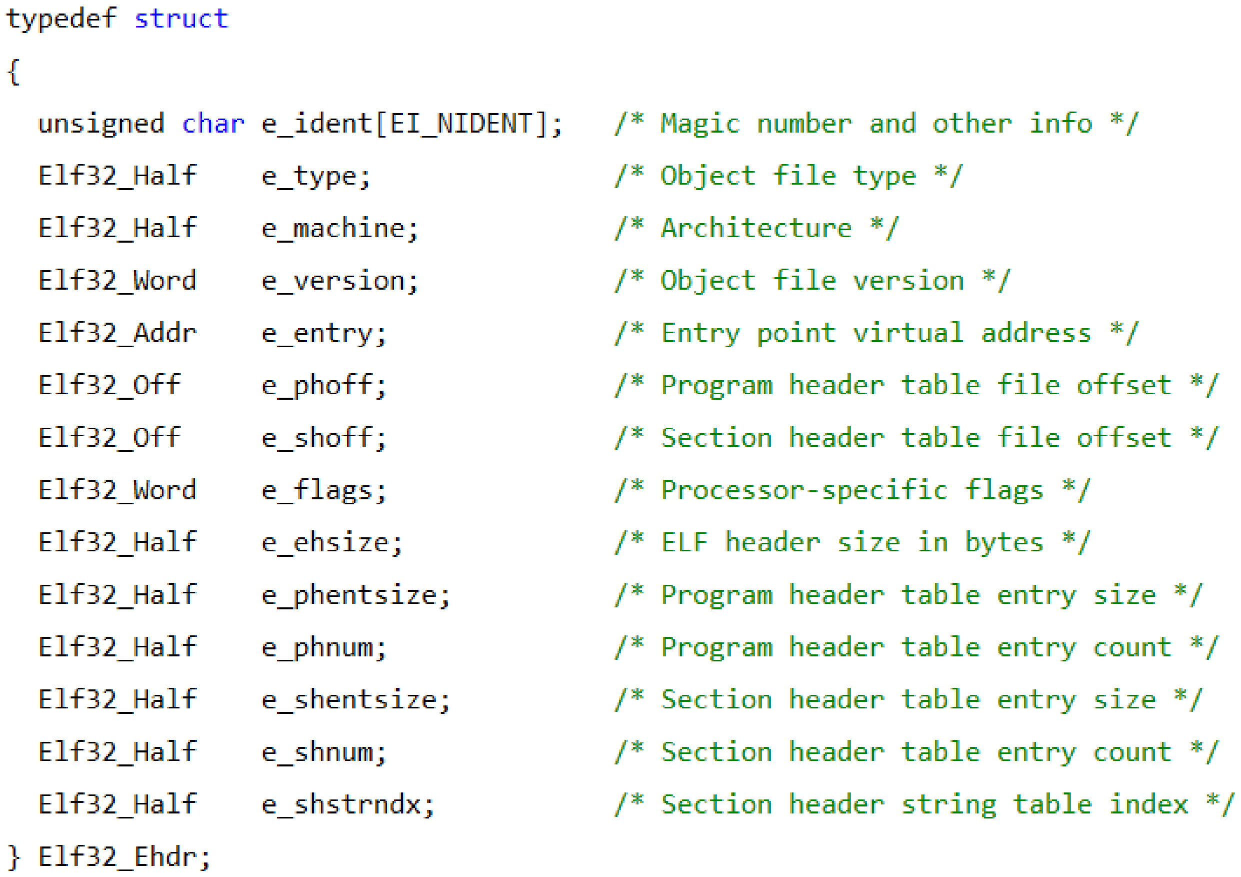
. . .

Section Header Table
optional

OSD1980

Project 1 – Bootloader

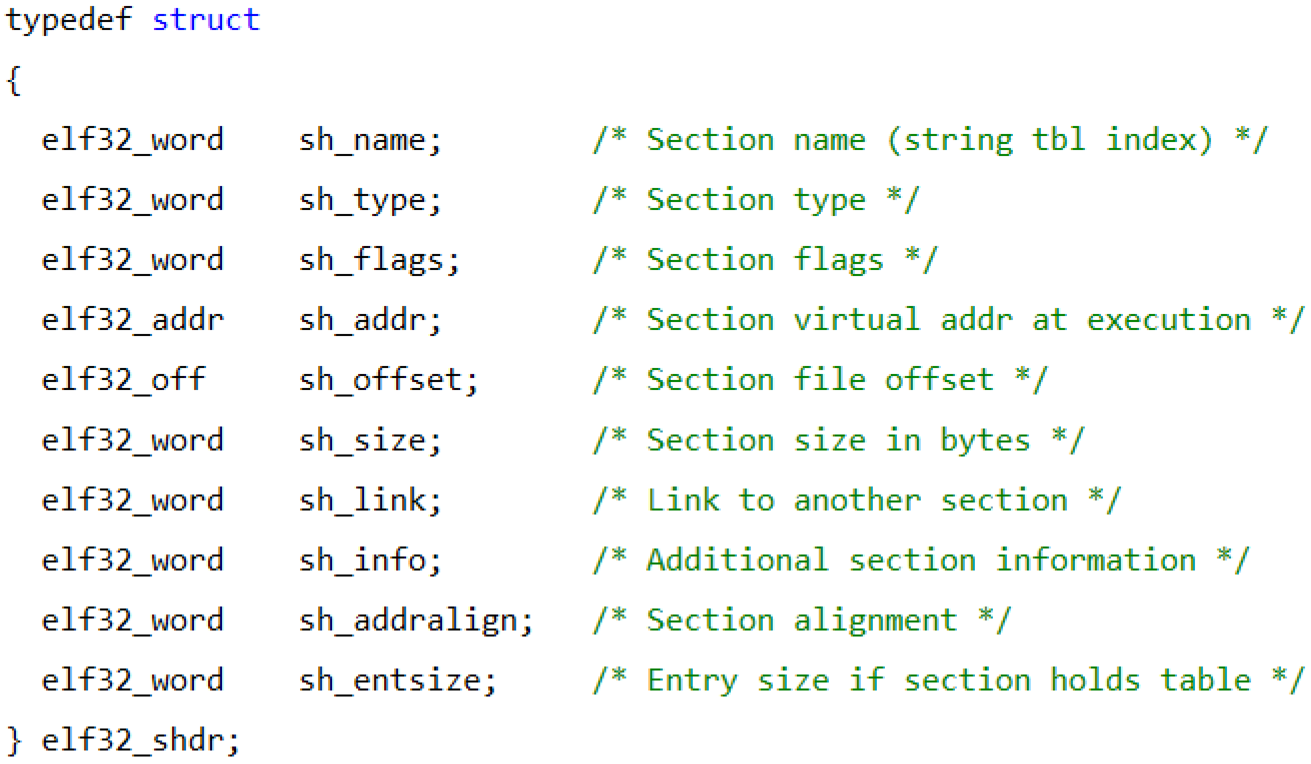
- ELF object file format
  - ELF header



```
typedef struct
{
  unsigned char e_ident[EI_NIDENT];   /* Magic number and other info */
  Elf32_Half    e_type;               /* Object file type */
  Elf32_Half    e_machine;            /* Architecture */
  Elf32_Word    e_version;            /* Object file version */
  Elf32_Addr    e_entry;              /* Entry point virtual address */
  Elf32_Off     e_phoff;              /* Program header table file offset */
  Elf32_Off     e_shoff;              /* Section header table file offset */
  Elf32_Word    e_flags;              /* Processor-specific flags */
  Elf32_Half    e_ehsize;             /* ELF header size in bytes */
  Elf32_Half    e_phentsize;          /* Program header table entry size */
  Elf32_Half    e_phnum;              /* Program header table entry count */
  Elf32_Half    e_shentsize;          /* Section header table entry size */
  Elf32_Half    e_shnum;              /* Section header table entry count */
  Elf32_Half    e_shstrndx;           /* Section header string table index */
} Elf32_Ehdr;
```

Project 1 – Bootloader

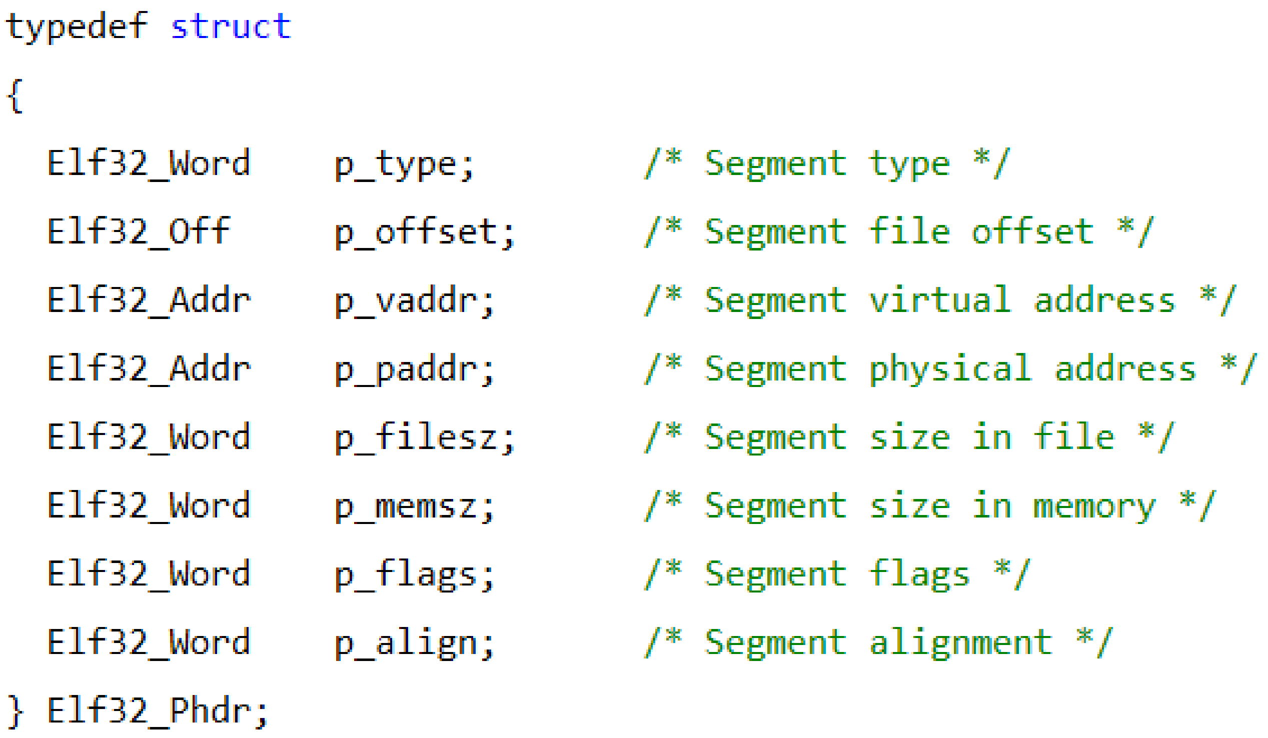
- ELF object file format
  - Section header



```
typedef struct
{
  elf32_word    sh_name;        /* Section name (string tbl index) */
  elf32_word    sh_type;        /* Section type */
  elf32_word    sh_flags;       /* Section flags */
  elf32_addr    sh_addr;        /* Section virtual addr at execution */
  elf32_off     sh_offset;      /* Section file offset */
  elf32_word    sh_size;        /* Section size in bytes */
  elf32_word    sh_link;        /* Link to another section */
  elf32_word    sh_info;        /* Additional section information */
  elf32_word    sh_addralign;   /* Section alignment */
  elf32_word    sh_entsize;     /* Entry size if section holds table */
} elf32_shdr;
```

Project 1 – Bootloader

- ELF object file format
  - Program header



```
typedef struct
{
  Elf32_Word    p_type;         /* Segment type */
  Elf32_Off     p_offset;       /* Segment file offset */
  Elf32_Addr    p_vaddr;        /* Segment virtual address */
  Elf32_Addr    p_paddr;        /* Segment physical address */
  Elf32_Word    p_filesz;       /* Segment size in file */
  Elf32_Word    p_memsz;        /* Segment size in memory */
  Elf32_Word    p_flags;        /* Segment flags */
  Elf32_Word    p_align;        /* Segment alignment */
} Elf32_Phdr;
```

Project 1 – Bootloader

- MIPS32 assembly language
  - 32 registers

Registers	Alternative name	Usage
$0	zero	Constant 0
$1	$at	Reserved by the assembler
$2 - $3	$v0 - $v1	Values from function results
$4 - $7	$a0 - $a3	Arguments, first four parameters for subroutine
$8 - $15	$t0 - $t7	Temporaries
$16 - $23	$s0 - $s7	Saved value
$24 - $25	$t8 - $t9	Temporaries

Project 1 – Bootloader

- MIPS32 assembly language
  - 32 registers

Registers	Alternative name	Usage
$26 - $27	$k0 - $k1	reserved for use by the interrupt/trap handler
$28	$gp	Global pointer
$29	$sp	Stack pointer
$30	$s8/$fp	Saved value / frame pointer
$31	$ra	Return address

Project 1 – Bootloader

- MIPS32 assembly language
  - Data types

    .ascii

    .byte: 8bit

    .half-word: 16bit

    .word: 32bit
  - a character requires 1 byte of storage
  - an integer requires 1 word of storage

Project 1 – Bootloader

- MIPS32 assembly language
  - Literals
    - numbers entered as is., e.g. 4
    - characters enclosed in single quotes. e.g. 'b'
    - strings enclosed in double quotes. e.g. "A string"

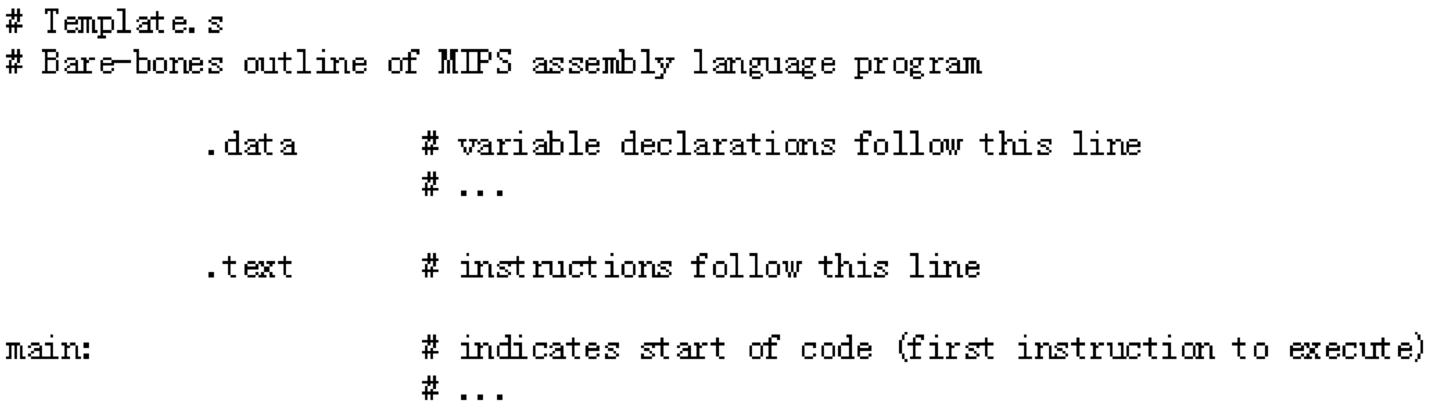
Project 1 – Bootloader

- MIPS32 assembly language
  - Assembler directives
    - Segment the program
  - .data
    - begins data segment
    - declares variable names used in program
    - storage allocated in main memory

```
name:  storage_type  values
val1:  .word  0x33
msg:  .ascii   “hello world\n”
```

Project 1 – Bootloader

- MIPS32 assembly language
  - .text: begins code segment, read-only, executable
    - contains instructions
    - starting point for code, e.g. given label main:



```
# Template.s
# Bare-bones outline of MIPS assembly language program

          .data       # variable declarations follow this line
                      # ...

          .text       # instructions follow this line

main:                 # indicates start of code (first instruction to execute)
                      # ...
```

Project 1 – Bootloader

- MIPS32 assembly language
  - RAM access
    - lw register_destination, RAM_source
    - lb register_destination, RAM_source
    - sw register_source, RAM_destination
    - sb register_source, RAM_destination
    - li register_destination, value

Project 1 – Bootloader

- MIPS32 assembly language
  - Indirect and Based Addressing
    - la $t0 val1
    - lw $t2, ($t0)
      - load word at RAM address contained in $t0 into $t2
    - sw $t2, ($t0)
      - store word in register $t2 into RAM at address contained in $t0
    - lw $t2, 4($t0)

Project 1 – Bootloader

- MIPS32 assembly language
  - Arithmetic instructions
    - add $t0,$t1,$t2
      - $t0 = $t1 + $t2
    - sub $t2,$t3,$t4
      - $t2 = $t3 - $t4
    - addi $t2,$t3, 5
      - $t2 = $t3 + 5

Project 1 – Bootloader

- MIPS32 assembly language
  - Control instructions – branches
    - beq $t0,$t1,target# branch to target if $t0 = $t1
    - blt $t0,$t1,target# branch to target if $t0 < $t1
    - ble $t0,$t1,target# branch to target if $t0 <= $t1
    - bgt $t0,$t1,target# branch to target if $t0 > $t1
    - bge $t0,$t1,target# branch to target if $t0 >= $t1
    - bne $t0,$t1,target# branch to target if $t0 <> $t1

Project 1 – Bootloader

- MIPS32 assembly language
  - Control instructions – jump
    - j target
      - Unconditional jump
    - jr $t3
      - Jump to address contained in $t3
    - jal sub_label
      - copy program counter to register $ra
      - jump to program statement at sub_label
    - jalr

Project 1 – Bootloader

- BIOS functions
  - Print character to serial port
    - address: 0xbfe4 8000
  - SSD_card_read
    - function address: 0x8007 b1a8
    - function parameters: address, offset, size

Project 1 – Bootloader

- Step by step
  - Task 1: setup the environment
  - Task 2: play with OpenLoongson SoC board to print characters
  - Task 3: given kernel and createimage, develop bootblock.s
  - Task 4: given kernel, develop your own createimage.c

Project 1 – Bootloader

- Requirement for design review
  - Answer following questions
    - Where do you place your bootblock
    - How to move kernel from disk to memory
    - Where do you place your kernel in the memory
    - Where is your kernel entry point
    - How to create disk image

Project 1 – Bootloader

- Requirement for developing
  - Finish following codes
    - Using SSD_card_read function to transfer kernel: 15
    - Given control to the kernel: 15
    - Create image: 20
    - Extended flag: 5
    - Kernel executes on VM: 5

Tips

- Learn to work on Linux
  - Watch out the outputs
- Read the task assignments carefully
- About inquiring TA
  - Think and Search before you ask TA
  - Discuss with your group mate/classmate
  - Try to describe your problem clearly
    - Pls. do not just show TA a screenshot