Overview

- This Lecture
  - Introduction & memory addressing
  - Source: ULK ch 1 & ch 2, ARM1176JZF-S Technical Reference Manual, etc
  - You can find them at http://www.cs.otago.ac.nz/cosc440/resources.php

Course objectives

- Discuss the design and research issues
- Examine the design and the internals of a real operating system (Linux & xv6);
- Enable the kernel programming skill
- Understand how an operating system interacts with modern CPU (ARM);

Course structure

- One two-hour lecture per week
- One two-hour guided lab per week
- Reading material (OSDI, SOSP), start now
- 100% internal assessment
  - Two programming assignments (20% and 30%)
  - One writing assignment (40%), including 10% presentation (ChatGPT is not allowed).
  - Q&A for 10 weeks starting from the second week (10%)

No Textbook

- Understanding the Linux kernel (3rd edition)
- Essential Linux Device Drivers
- Linux Device Drivers (3rd edition)
  - Code examples are a bit obsolete
- Writing Linux Device Drivers (good for lab)
- Linux source cross reference
  - http://lxr.linux.no
- Google or ChatGPT (may give you some answers but you need to check their correctness).
- http://www.kernel.org/doc/htmldocs/

Why OS?

- OS is useful for application programming
  - Programming on H/W is painful
  - OS enables portability, multi-programming, and standard utility.
- OS kernel (excl. the system tools in u/s)
  - A H/W management library
  - Abstract layer on H/W with better properties
  - Layers: H/W—kernel—user
- Crown of programming
- Skills are applicable to embedded systems

OS kernel design

- The design cares a lot about interfaces and the kernel internal structure
- It demands real-time processing
  - Programmers should have awareness of time when programming.
- The kernel typically provides
  - Processes, memory, file system, device drivers, interprocess communication, user management, security policy, access control, time, etc.

Good kernel design

- Should abstract the H/W for convenience
  - The API is via system calls like open(), read().
- Should multiplex the hardware among multiple applications/users
- Should isolate applications to contain bugs
- Should allow sharing and communication among applications

Why OS design interesting?

- Fast vs abstract
- Performance vs portability
- Many features vs few mechanisms
- Convenience vs composibility
  - Fork() & exec() vs create_process()
- Open problems: security, multi-core

Multi-user OS

- Protection and isolation
- Users and groups
- Files are protected by access rights according to user id and group id
- Processes differ according to user id and group id
- Fair sharing of resources

Processes

- Users share resources fairly using processes
- A process virtualizes CPU
  - Uses process control block for each process in kernel (*task_struct* in Linux, proc.h in xv6)
- Process scheduling
  - Preemptive and non-preemptive
  - Round robin and etc.
  - Process states: running, ready, blocked, etc
  - Refer to proc.c in xv6
- All processes form a tree

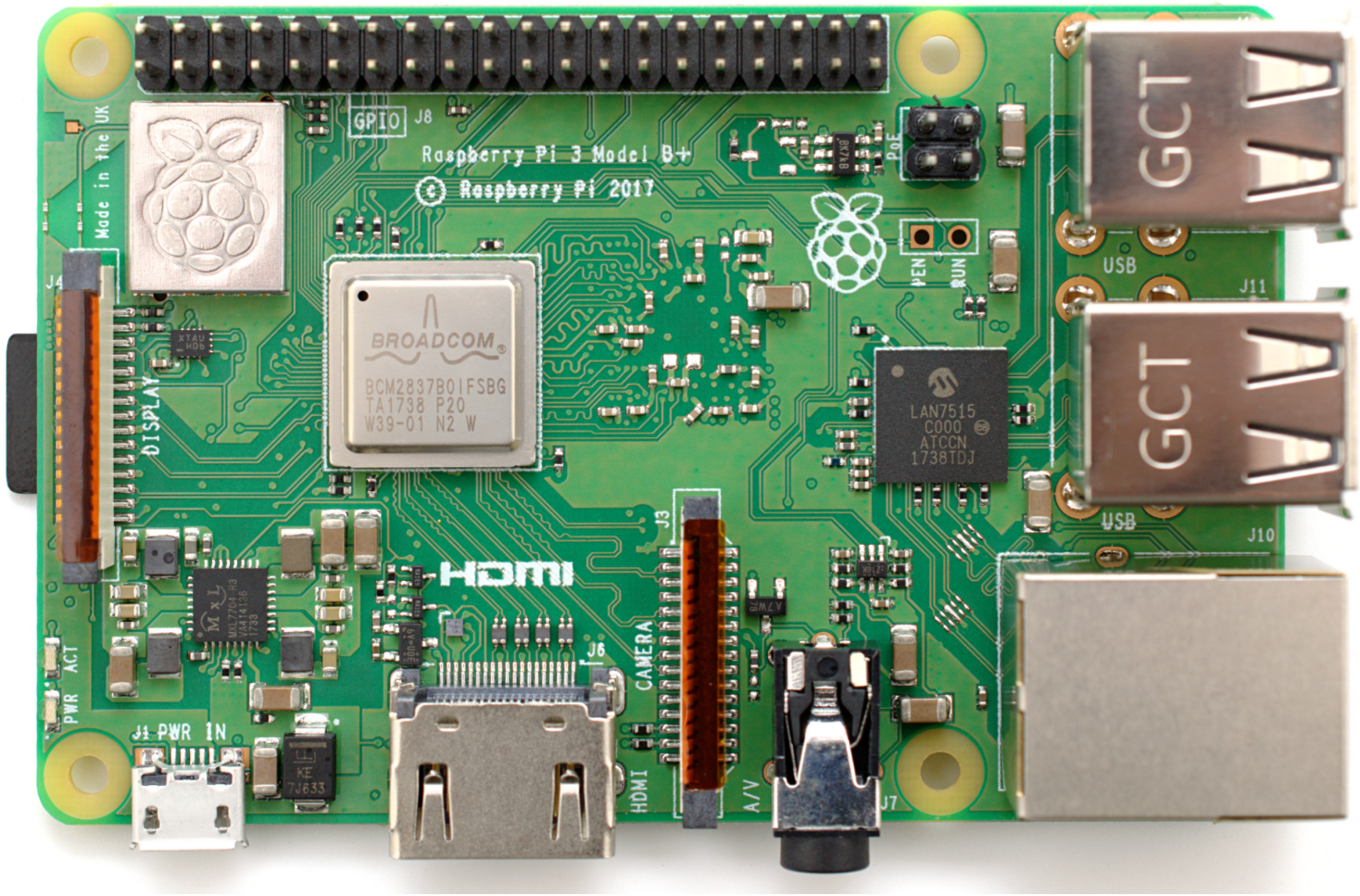
System calls

- OS API for user processes
- Open(), read(), write(), ioctl(), close()
  - Use them in Lab 1 to implement a *cat* program.
- Fork(), exit(), wait(), exec(), kill()
- Getpid(), sleep(), sbrk(), dup(), pipe()
- Chdir(), mkdir(), mknod(), fstat(), link(), unlink()

Raspberry Pi

- Broadcom BCM2837 SoC
  - ARM Cortex-A53 1.2 GHz processor (ARMv8)
  - VideoCore GPU, 1 GB RAM
- SD card for booting and storage
- 2.5W to 3.5W
- Audio/video outputs, video input for camera
- HDMI, USB 2.0 (Ethernet), 40 GPIO, UART, I2C, etc
- Has Linux, FreeBSD, Plan9, xv6 etc

The board


GPIO
J8
Raspberry Pi 3 Model B+
© Raspberry Pi 2017
Made in the UK
BROADCOM
BCM2837B0IFSBG
TA1738 P20
W39-01 N2 W
LAN7515
C000
ATCCN
1738TDJ
GCT
GCT
USB
USB
J11
J10
PoE
PEN
RUN
DISPLAY
J4
HDMI
J3
CAMERA
J6
HDMI
A/V
J7
PWR ACT
J1 PWR IN

Implemented ideas with Pi

- Home automation system
- Wall mounted voice controlled screen
- Security webcam with motion sensor
- Control sprinkler system
- Game server
- An alarm system
- MP3 player
- Medical input device shield
- Supercomputer
- Voice activated coffee machine
- Make old TV into a Smart TV
- …

xv6

- A modern implementation of Sixth Edition Unix in ANSI C for multiprocessor x86 systems
  - The code was only available on PDP-11
- Used for pedagogical purposes at MIT
  - Run in simulated environment like QEMU
  - I made it PC-bootable and start use the code for COSC440 from 2013.
- Well documented due to Lions' Commentary on UNIX 6th Edition, with Source Code.
- It has process management, memory management, RAM FS, device drivers such as console

Why port xv6 to RPI?

- Existing OS like Linux is too complex for students
- Run xv6 in simulator (QEMU) is boring.
- We don't have PCs available in the department
  - Old PCs are very frail.
- ARM-based SoC is more popular
  - Embedded programming on SoC is in demand
- RPI is cheap and popular
  - $40, 30 millions sold out since 2012

ARM CPU

- ARM is based on RISC architecture. ARMv8 ISA is used in Raspberry Pi (RPI)
- CPU runs instructions continuously unless redirected by instructions or interrupted by events
- General-purpose registers
  - R0-R15, R13(SP), R14(LR), R15(PC)
- Program status registers
  - CPSR, SPSR

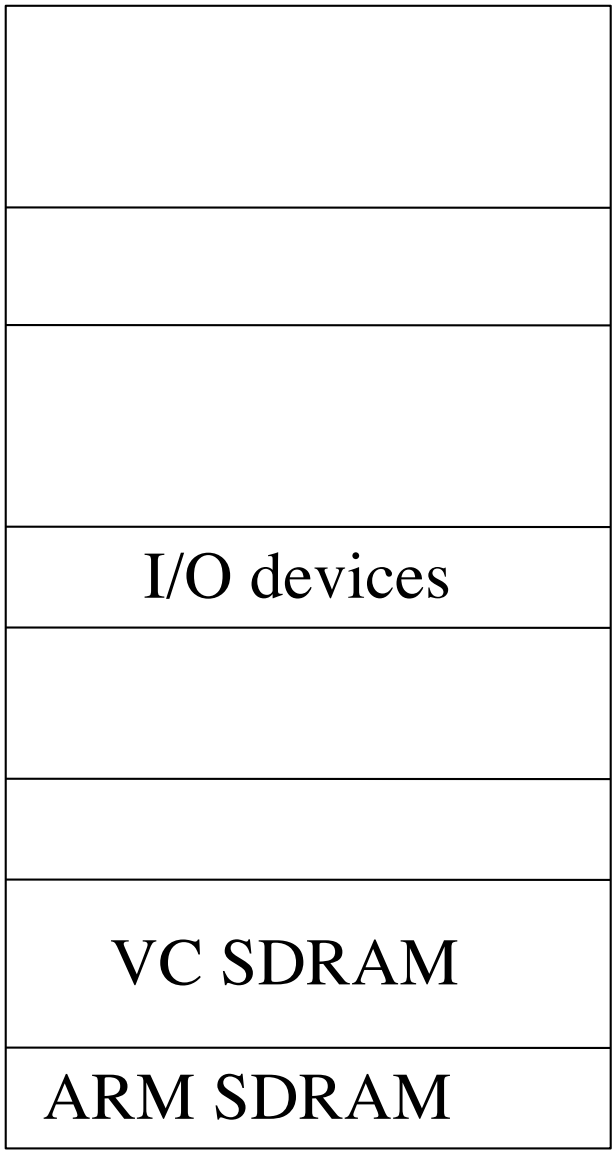
ARM CPU (cont.)

- Instructions are stored in memory
  - They are fetched into CPU for execution through instruction pointer (a register called PC, aka. R15)
  - PC is incremented after each instruction is completed unless
  - It is modified by branch instructions like *bx* and *bl*
  - Load/store registers: ldr, str,
  - Arithmetic instructions: add, sub, mul, shift
  - Special instructions for special registers

Memory-Mapped I/O

- Use physical memory addresses for I/O
  - No size limit for I/O address space
  - No need of special I/O instructions
  - Routing to appropriate device through system controller
- Behave differently from normal memory
  - Reads/writes could have side effects
  - Results of reads could change due to ext. events
- Direct Memory Access (DMA)

Physical memory map



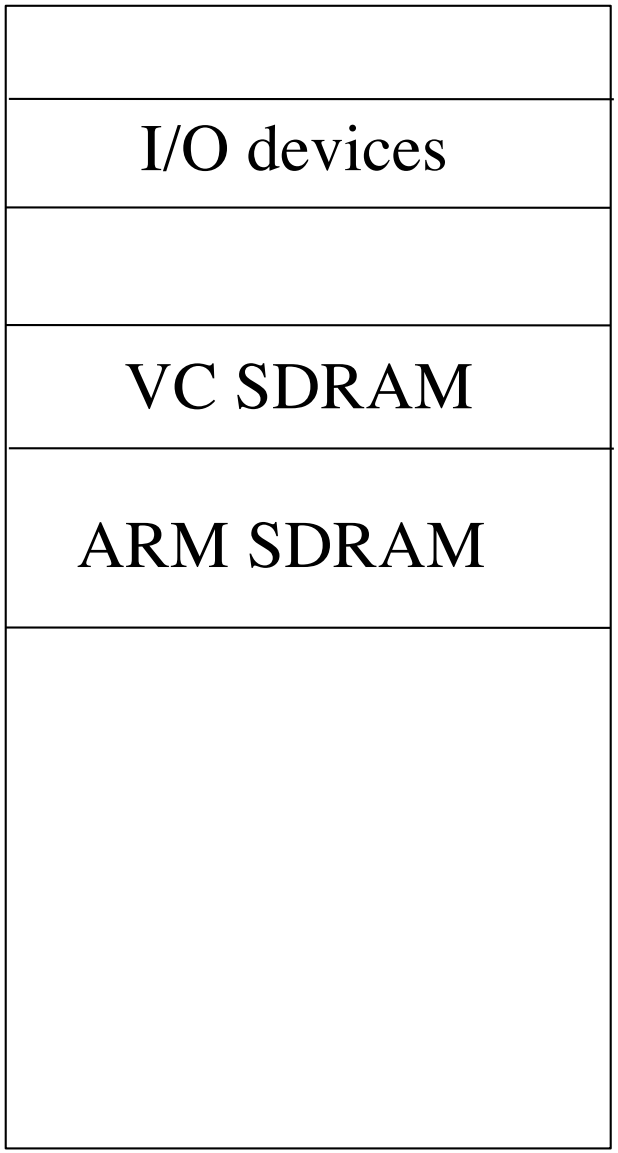
0x40000000(512MB)

0x20000000 – 0x20ffffff(16MB)

0xc000000 – 0xffffffff (256MB)

0x00000000 – 0xbffffff (192MB)

Virtual memory map



0xffffffff(4GB)

0xfe000000

0xc0000000 – user/kernel separation

User-mode page-mapped virtual addresses

0x00000000

ARM stack manipulation

- push r0
  - sub sp, #4
  - str r0, [sp]
- pop r0
  - ldr r0, [sp]
  - add sp, #4

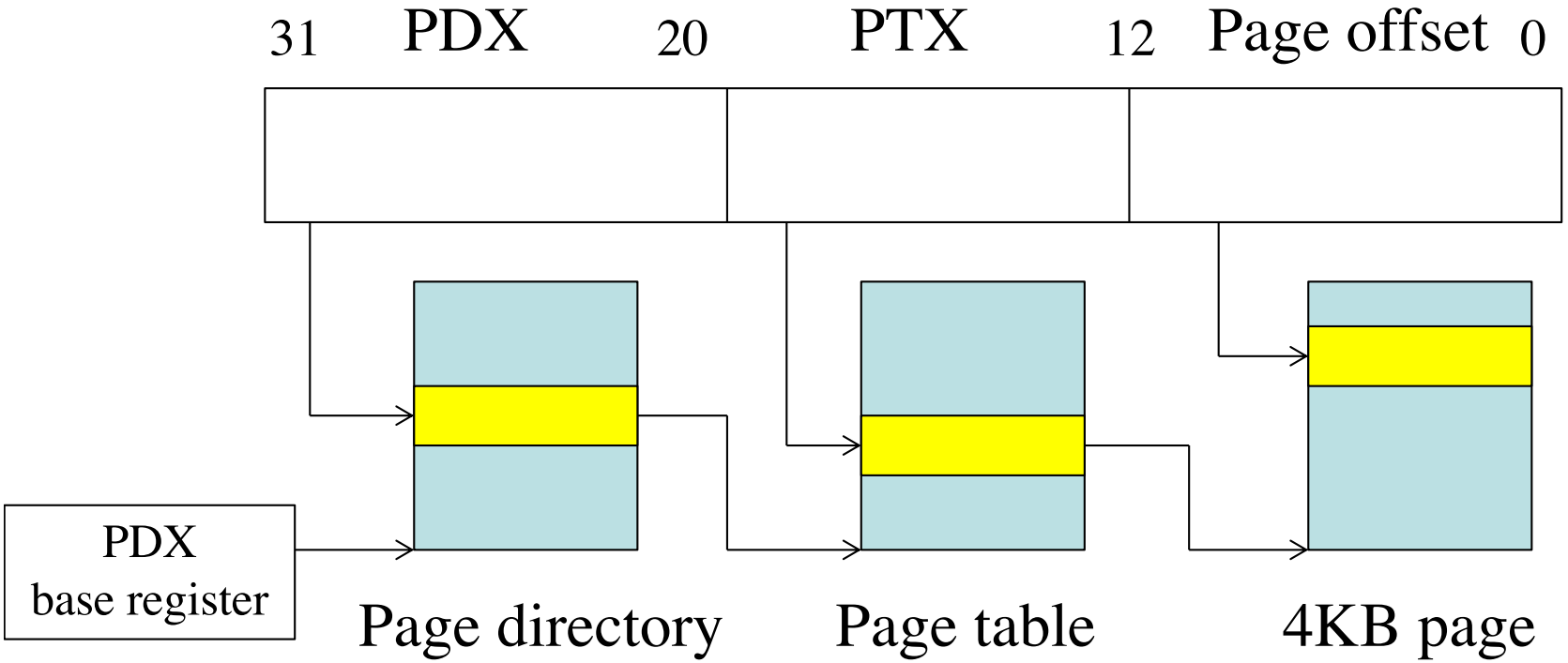
ARM GCC calling convention

- Call a function
  - *bl 0x12345*
    - mov lr, pc; mov pc, #0x12345
- Return from a function
  - *movs pc, lr* or *pop pc* if lr is stored in the stack.
- Parameters are passed with registers
  - r0, r1, r2, r3, aka. caller saved registers
  - More parameters then the stack will be used
  - Results are put into r0 and r1
- Callee saved registers: r4-r11

Memory protection

- ARMv8 has 16 domains for memory security, though we only use one domain in xv6
- Each page can be set no-access, read-only or read/write for kernel or user mode.
- Each page can be set cacheable or not
  - Region attributes: Strongly Ordered, Device, cacheable Write-Through, and cacheable Write-Back.
- Page sizes: 4KB, 64KB, 1MB, 16MB

ARM paging structure



Each entry needs 4 bytes

Four pages are needed for PDX

A quarter page is used for PTX

Translation Lookaside Buffers (TLB):

Hardware cache to speed up linear address translation

Linux linear address space

- Kernel/user space split
  - 1G/3G split in 32-bit
  - But at 0xffff880000000000 in 64-bit
  - 0xc0000000 and above for the kernel
  - Below 0xc0000000 for user processes
- Linear/physical mapping layout
  - 896M for straight mapping between kernel linear space and physical memory
  - 128M (896M-1G) reserved for dynamic mapping of high memory and virtual memory

Questions on course material as an assessment of study

- From week 2, you should send me questions on the lecture and lab of the week before every Tuesday morning, for 10 weeks.
- Depending on the quality of the questions and the communications at the lecture, you will be given a maximum mark of 1.
- If you don't attend a lecture without a valid reason, you will lose the mark.

Readings

- Have a look at OSDI/SOSP websites and choose the papers you are interested to read.