2011-07-07

2011年 前学期 TOKYO TECH

計算機アーキテクチャ 第一 (E)

仮想記憶

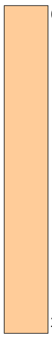
吉瀬 謙二　計算工学専攻
kise_at_cs.titech.ac.jp
W641講義室　木曜日13:20 － 14:50

Acknowledgement

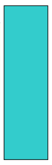
- Lecture slides for Computer Organization and Design, Third Edition, courtesy of **Professor Mary Jane Irwin**, Penn State University
- Lecture slides for Computer Organization and Design, third edition, Chapters 1-9, courtesy of **Professor Tod Amon,** Southern Utah University.

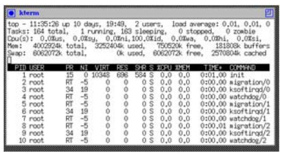
例：32ビット（4GB）のメモリ空間

0x00000000　00000000 00000000 00000000 00000000_2 = 0_{10}



2GB Memory !





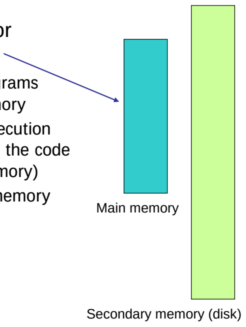
```
top - 11:35:26 up 10 days, 19:49,  2 users,  load average: 0.01, 0.01, 0
Tasks: 164 total,   1 running, 163 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,
Mem:   4002924k total,  3252404k used,   750520k free,   181808k buffers
Swap:  6062072k total,        0k used,  6062072k free,  2570804k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    1 root      15   0 10348  696  584 S  0.0  0.0   0:01.00 init
    2 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/0
    4 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    5 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/1
    6 root      34  19     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/1
    7 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/1
    8 root      RT  -5     0    0    0 S  0.0  0.0   0:00.01 migration/2
    9 root      34  19     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/2
   10 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/2
```

0xFFFFFFFF　11111111 11111111 11111111 11111111_2 = 4,294,967,296 - 1_{10}

Virtual Memory（仮想記憶）

- Use main memory as a “**cache**” for secondary memory
  - Provides the ability to easily run programs **larger** than the size of physical memory
  - **Simplifies** loading a program for execution by providing for code relocation (i.e., the code can be loaded anywhere in main memory)
  - Allows efficient and safe sharing of memory among **multiple programs**

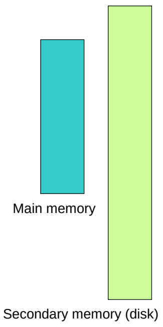


Virtual Memory（仮想記憶）

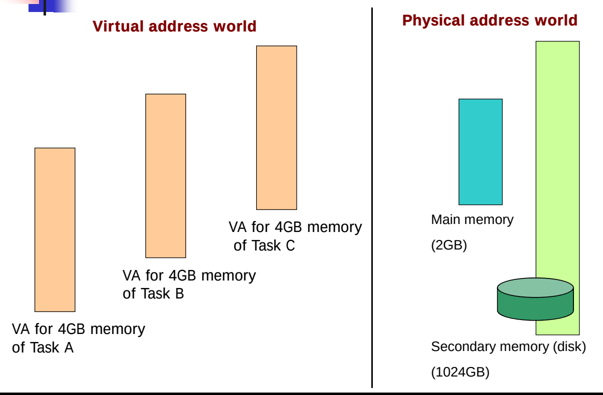
- What makes it work? – again the **Principle of Locality**
  - A program is likely to access a relatively small portion of its address space during any period of time

Virtual Memory（仮想記憶）

- Each program is compiled into its own address space – a “virtual address (VA)” space
- Physical address (PA) for the access of physical devices
  - During run-time each **virtual address, VA**（仮想アドレス）must be translated to a **physical address, PA**（物理アドレス）

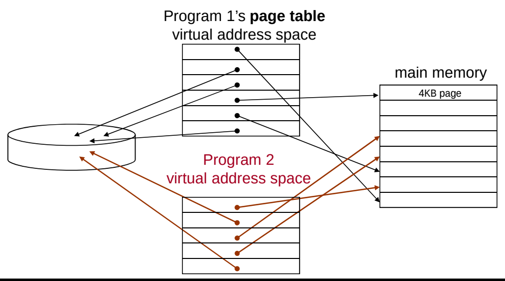


Virtual Memory（仮想記憶）



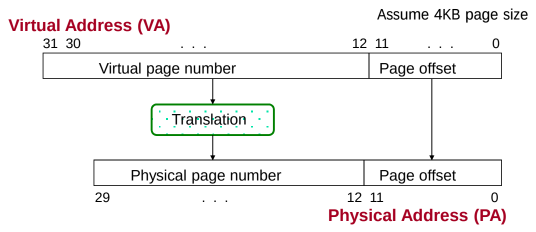
Two Programs Sharing Physical Memory

- A program's address space is divided into **pages** (all one fixed size) or **segments** (variable sizes)
  - The starting location of each page (either in **main memory** or in **secondary memory**) is contained in the program's **page table**



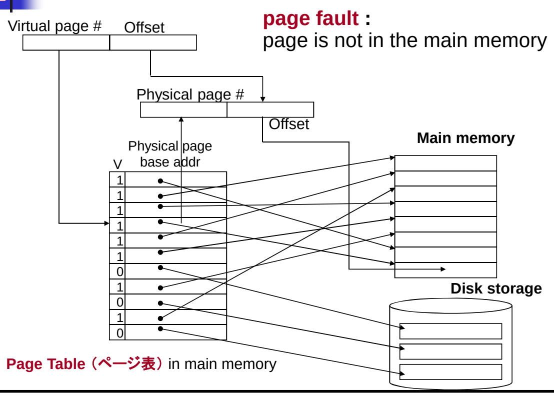
Address Translation

- A virtual address is translated to a physical address by a combination of hardware and software

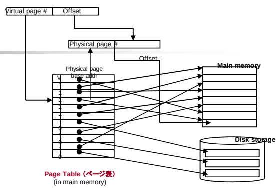


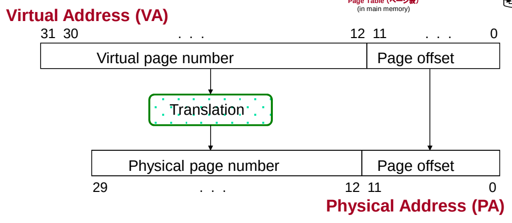
- So each memory request **first** requires an **address translation** from the virtual space to the physical space

Address Translation Mechanisms



Address Translation

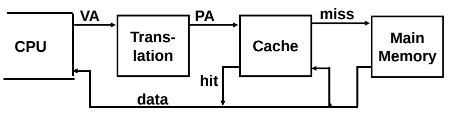




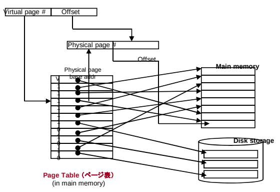
- ページサイズ4KBの場合，シンプルなページ表のメモリサイズは？

Virtual Addressing with a Cache

- Thus it takes an ***extra*** memory access to translate a virtual address to a physical address



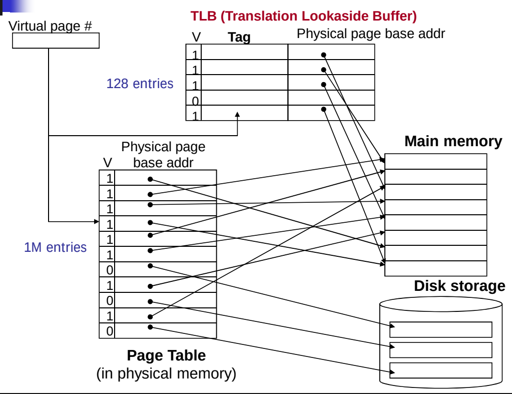
- This makes memory (cache) accesses **very expensive** (if every access was really ***two*** accesses)



Virtual Addressing, the hardware fix

- The hardware fix is to use a **Translation Lookaside Buffer (TLB)**（アドレス変換バッファ）
  - a small cache that keeps track of recently used address mappings to avoid having to do a page table lookup

Making Address Translation Fast



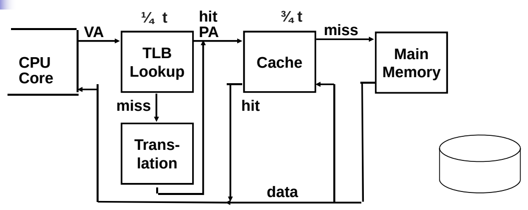
Translation Lookaside Buffers (TLBs)

- Just like any other cache, the TLB can be organized as fully associative, set associative, or direct mapped

V	Virtual Page #	Physical Page #			

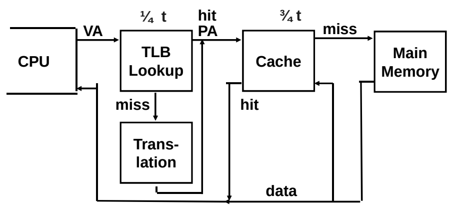
- TLB access time is **typically smaller** than cache access time (because TLBs are much smaller than caches)
  - TLBs are typically not more than 128 to 256 entries even on high end machines

A TLB in the Memory Hierarchy



- **A TLB miss** – is it a page fault or a TLB miss ?
  - If the page is in main memory, then the TLB miss can be handled (in hardware or software) by loading the translation information from the page table into the TLB
    - Takes 10's of cycles to find and load the translation info into the TLB
  - If the page is not in main memory, then it's a true page fault
    - Takes 1,000,000's of cycles to service a page fault

A TLB in the Memory Hierarchy



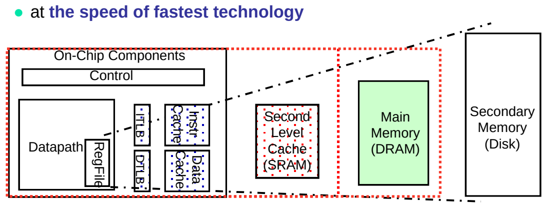
- **page fault** : page is not in physical memory
- **TLB misses** are much more frequent than true page faults

Two Machines' TLB Parameters

	Intel P4	AMD Opteron
TLB organization	1 TLB for instructions and 1TLB for data Both 4-way set associative Both use ~LRU replacement Both have 128 entries TLB misses handled in hardware	2 TLBs for instructions and 2 TLBs for data Both L1 TLBs fully associative with ~LRU replacement Both L2 TLBs are 4-way set associative with round-robin LRU Both L1 TLBs have 40 entries Both L2 TLBs have 512 entries TBL misses handled in hardware

A Typical Memory Hierarchy

- By taking advantage of **the principle of locality**（局所性）
  - Present **much memory** in **the cheapest technology**
  - at **the speed of fastest technology**



Speed (%cycles):	½'s	1's	10's	100's	1,000's
Size (bytes):	100's	K's	10K's	M's	G's to T's
Cost:	highest				lowest

The Hardware/Software Boundary

- What parts of the virtual to physical address translation is done by or assisted by the hardware?
  - **Translation Lookaside Buffer (TLB)** that caches the recent translations
    - TLB access time is part of the cache hit time
    - May cause an extra stage in the pipeline for TLB access
  - Page table storage, fault detection and updating
    - **Page faults** result in **interrupts (precise)** that are then handled by the **OS**
    - Hardware must support (i.e., update appropriately) **Dirty** and **Reference bits (e.g., ~LRU)** in the Page Tables

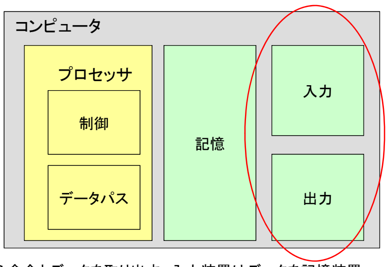
2011-07-07 2011年 前学期 TOKYO TECH

計算機アーキテクチャ 第一 (E)

入出力制御，割り込み

吉瀬 謙二　計算工学専攻
kise_at_cs.titech.ac.jp
W641講義室　木曜日13:20 ー 14:50

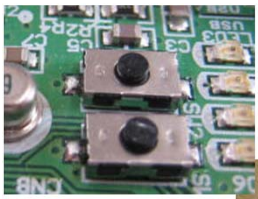
コンピュータ(ハードウェア)の古典的な要素

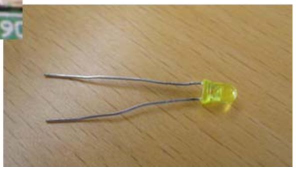


プロセッサは記憶装置から命令とデータを取り出す。入力装置はデータを記憶装置に書き込む。出力装置は記憶装置からデータを読みだす。制御装置は、データパス、記憶装置、入力装置、そして出力装置の動作を指定する信号を送る。

出典：パターソン ＆ ヘネシー、コンピュータの構成と設計 22

Input and Output Devices（入出力装置）





23

Input and Output Devices（入出力装置）

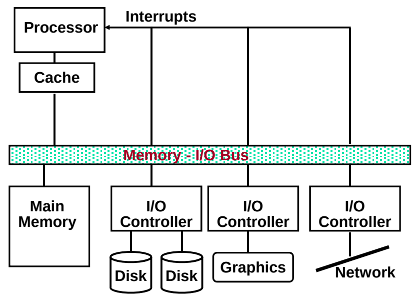
- I/O devices are **diverse** with respect to
  - **Behavior**（動作）– input, output or storage
  - **Partner**（相手）– human or machine
  - **Data rate**（転送速度）– the peak rate at which data can be transferred between the I/O device and the main memory or CPU

Device	Behavior	Partner	Data rate (Mb/s)
Keyboard	input	human	0.0001
Mouse	input	human	0.0038
Laser printer	output	human	3.2000
Graphics display	output	human	800.0000-8000.0000
Network/LAN	input or output	machine	100.0000-1000.0000
Magnetic disk	storage	machine	240.0000-2560.0000

8 orders of magnitude range

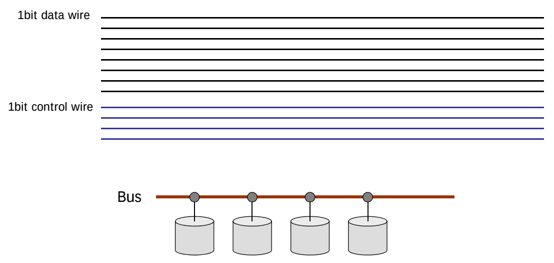
24

A Typical I/O System（代表的な入出力装置）



Bus, I/O System Interconnect

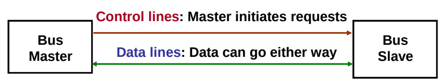
- A **bus**（バス）is a **shared** communication link
  (a single set of wires used to connect multiple subsystems)



Bus, I/O System Interconnect

- A **bus**（バス）is a shared communication link (a single set of wires used to connect multiple subsystems)
  - **Advantages**
    - **Low cost** – a single set of wires is shared in multiple ways
    - **Versatile**（**多目的**）– new devices can be added easily and can be moved between computer systems that use the same **bus standard**
  - **Disadvantages**
    - Creates a communication bottleneck – **bus bandwidth** limits the maximum **I/O throughput**
- The maximum bus speed is largely limited by
  - The **length** of the bus
  - The **number** of devices on the bus

Bus Characteristics

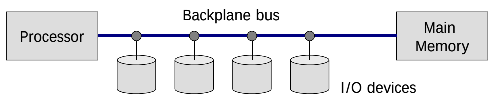

Control lines: Master initiates requests
Bus Master
Data lines: Data can go either way
Bus Slave

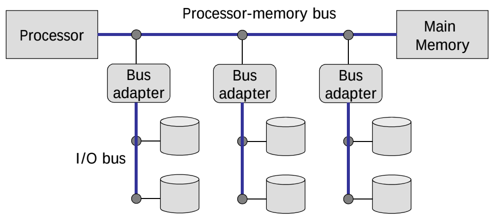
- **Control lines**
  - Signal requests and acknowledgments
  - Indicate what type of information is on the data lines
- **Data lines**
  - Data, addresses, and complex commands
- **Bus transaction** consists of
  - Master issuing the command (and address) – request
  - Slave receiving (or sending) the data – action
  - *Defined by what the transaction does to memory*
    - **Input** – inputs data from the I/O device to the memory
    - **Output** – outputs data from the memory to the I/O device

Types of Buses

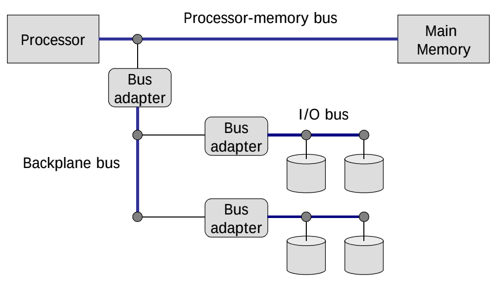
- **Processor-memory bus**
  - Short and high speed
  - Matched to the memory system to maximize the memory-processor bandwidth
  - Optimized for cache block transfers
- **I/O bus** (industry standard, e.g., SCSI, USB, Firewire)
  - Usually is lengthy and slower
  - Needs to accommodate a wide range of I/O devices
  - Connects to the processor-memory bus or backplane bus
- **Backplane bus** (industry standard, e.g., ATA, PCIexpress)
  - The backplane is an interconnection structure within the chassis
  - Used as an intermediary bus connecting I/O busses to the processor-memory bus

Types of Buses





Types of Buses



31

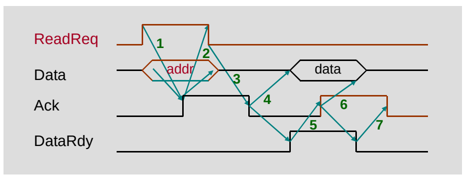
Synchronous(同期式), Asynchronous(非同期式) Buses

- **Synchronous bus** (e.g., processor-memory buses)
  - Includes a clock in the control lines and has a fixed protocol for communication that is **relative** to the clock
  - **Advantage**: involves very little logic and can run very fast
  - **Disadvantages**:
    - Every device communicating on the bus must use same clock rate
    - To avoid **clock skew**, they cannot be long if they are fast
- **Asynchronous bus** (e.g., I/O buses)
  - It is not clocked, so requires a **handshaking protocol** and additional control lines (**ReadReq, Ack, DataRdy**)
  - **Advantages**:
    - Can accommodate a wide range of devices and device speeds
    - Can be lengthened without worrying about clock skew or synchronization problems
  - **Disadvantage**: slow

32

Asynchronous Bus **Handshaking Protocol**

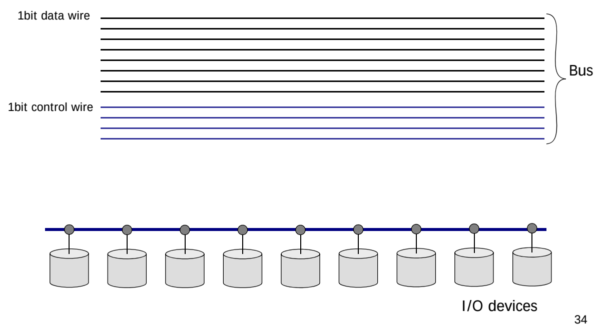
An I/O device reads data from memory.



1. Memory sees **ReadReq**, reads **addr** from data lines, and raises **Ack**
2. I/O device sees **Ack** and releases the **ReadReq** and data lines
3. Memory sees **ReadReq** go low and drops Ack
4. When memory has data ready, it places it on data lines and raises **DataRdy**
5. I/O device sees **DataRdy**, reads the data from data lines, and raises **Ack**
6. Memory sees **Ack**, releases the data lines, and drops **DataRdy**
7. I/O device sees **DataRdy** go low and drops **Ack**

33

The Need for Bus **Arbitration** (調停)



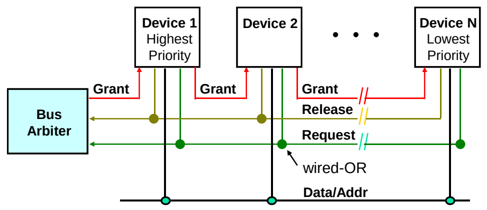
34

The Need for Bus **Arbitration** (調停)

- Multiple devices may need to use the bus **at the same time**
- **Bus arbitration schemes** usually try to balance:
  - **Bus priority** – the highest priority device should be serviced first
  - **Fairness** – even the lowest priority device should never be completely locked out from the bus
- **Bus arbitration schemes** can be divided into four classes
  - **Daisy chain arbitration**
  - **Centralized, parallel arbitration**
  - Distributed arbitration by collision detection
    - device uses the bus when its not busy and if a collision happens (because some other device also decides to use the bus) then the device tries again later (Ethernet)
  - Distributed arbitration by self-selection

35

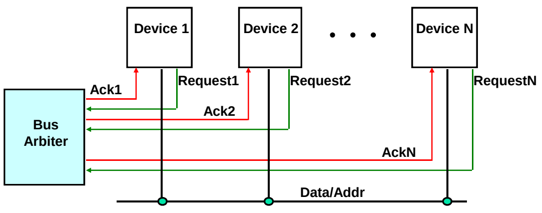
Daisy Chain Bus Arbitration (デイジーチェイン方式)



- **Advantage:** simple
- **Disadvantages**:
  - Cannot assure fairness – a low-priority device may be locked out
  - Slower – the daisy chain grant signal limits the bus speed

36

Centralized Parallel Arbitration（集中並列方式）



- **Advantages**: flexible, can assure fairness
- **Disadvantages**: more complicated arbiter hardware
- Used in essentially all processor-memory buses and in high-speed I/O buses

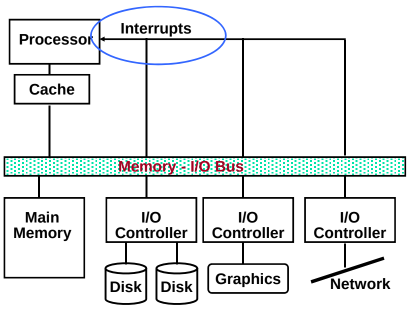
37

The Need for Bus **Arbitration**（調停）

- Multiple devices may need to use the bus **at the same time**
- **Bus arbitration schemes** usually try to balance:
  - Bus priority – the highest priority device should be serviced first
  - Fairness – even the lowest priority device should never be completely locked out from the bus
- **Bus arbitration schemes** can be divided into four classes
  - **Daisy chain arbitration**
  - **Centralized, parallel arbitration**
  - Distributed arbitration by collision detection（分散衝突検出方式）
    - device uses the bus when its not busy and if a collision happens (because some other device also decides to use the bus) then the device tries again later (**Ethernet**)
  - Distributed arbitration by self-selection（分散型自己判定方式）

38

I/O Systemの利用方法と割り込み



39

Communication of I/O Devices and Processor

- How the processor directs the I/O devices
  - **Memory-mapped I/O**
    - Portions of the high-order memory address space are assigned to each I/O device
    - Read and writes to those memory addresses are interpreted
      as commands to the I/O devices
    - Load/stores to the I/O address space can only be done by the OS
  - **Special I/O instructions**

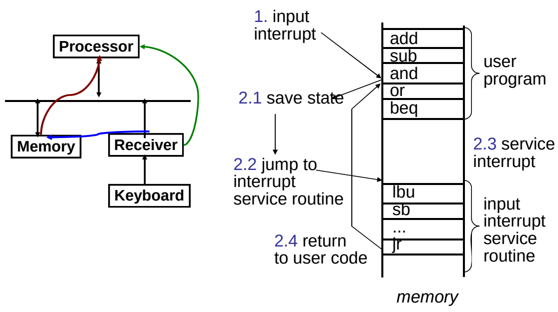
40

Communication of I/O Devices and Processor

- How the I/O device communicates with the processor
  - **Polling** – the processor periodically checks the status of an I/O device to determine its need for service
    - Processor is totally in control – but does **all** the work
    - Can waste a lot of processor time due to speed differences
  - **Interrupt-driven I/O** – **the I/O device issues an interrupts to the processor to indicate that it needs attention**

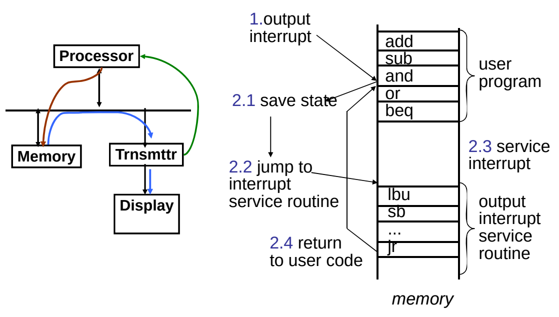
41

Interrupt-Driven Input



42

Interrupt-Driven Output



43

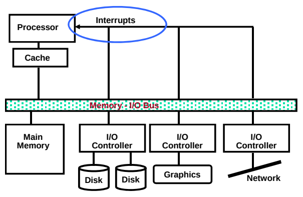
Interrupt-Driven I/O

- An I/O interrupt is **asynchronous**
  - Is not associated with any instruction so doesn't prevent any instruction from completing
    - You can pick your own convenient point to handle the interrupt
- With I/O interrupts
  - Need a way to identify the device generating the interrupt
  - Can have different urgencies (so may need to be prioritized)
- **Advantages** of using interrupts
  - No need to continuously poll for an I/O event; user program progress is only suspended during the actual transfer of I/O data to/from user memory space
- **Disadvantages** – special hardware is needed to
  - Cause an interrupt (I/O device) and detect an interrupt and save the necessary information to resume normal processing after servicing the interrupt (processor)

44

Direct Memory Access (DMA)

- For high-bandwidth devices (like disks) **interrupt-driven I/O** would consume a *lot* of processor cycles
- **DMA** – the I/O controller has the ability to transfer data **directly** to/from the memory without involving the processor
- There may be multiple DMA devices in one system



45

Direct Memory Access (DMA) how to?

1. The processor initiates the DMA transfer by supplying the I/O device address, the operation to be performed, the memory address destination/source, the number of bytes to transfer
2. The I/O DMA controller manages the entire transfer (possibly thousand of bytes in length), arbitrating for the bus
3. When the DMA transfer is complete, the I/O controller interrupts the processor to let it know that the transfer is complete

46

I/O and the **Operating System**

- The operating system acts as the interface between the I/O hardware and the program requesting I/O
  - To protect the **shared I/O resources**, the user program is not allowed to communicate directly with the I/O device
- Thus **OS** must be able to give commands to I/O devices, handle interrupts generated by I/O devices, provide fair access to the shared I/O resources, and schedule I/O requests to enhance system throughput
  - I/O interrupts result in a transfer of processor control to the **supervisor (OS) process**

47

参考書

- **コンピュータの構成と設計 第3版**、パターソン＆ヘネシー（成田光彰　訳）、日経BP社、2006
- コンピュータアーキテクチャ 定量的アプローチ 第4版
翔泳社, 2008
- コンピュータアーキテクチャ,
村岡 洋一 著, 近代科学社, 1989
- 計算機システム工学,
富田 真治, 村上 和彰 著, 昭晃堂, 1988
- コンピュータハードウエア,
富田 真治, 中島 浩 著, 昭晃堂, 1995
- 計算機アーキテクチャ,
橋本 昭洋 著, 昭晃堂, 1995

48

参考書

- **コンピュータの構成と設計 第3版**、パターソン＆ヘネシー（成田光彰　訳）、日経BP社、2006
- **コンピュータアーキテクチャ 定量的アプローチ 第4版**
  翔泳社, 2008
- コンピュータアーキテクチャ,
  村岡 洋一 著, 近代科学社, 1989
- 計算機システム工学,
  富田 真治, 村上 和彦 著, 昭晃堂, 1988
- コンピュータハードウエア,
  富田 真治, 中島 浩 著, 昭晃堂, 1995
- 計算機アーキテクチャ,
  橋本 昭洋 著, 昭晃堂, 1995

ヘネシー&パターソン
コンピュータアーキテクチャ
定量的アプローチ 第4版
Computer Architecture: A Quantitative Approach 4/E
John L. Hennessy, David A. Patterson

Computer Architecture & Design

期末試験

- 期末試験
  - 2011年07月21日(木) W641講義室, 5,6時限