Embedded sys

① Basic Concepts and Computer Evolution

Organization and Architecture

Computer architecture :
attribute that have direct impact on the logical execution
Computer organization:
operational units + interconnections , realize the arch specifications
Instruction set Architecture :
structure of a computer
programmer must understand to write a correct program for that machine
machine description
hardware designer must understand to design a correct implementation of the computer

structure and function

Function :
The operation of individual components as part of the structure
1) data processing
2) data storage
3) data movement
4) control
Structure : ~mumw
single core : CPU , Main memory , I/O , System Interconnection
↳ control Unit , ALU, Registers , CPU Interconnection
↳ sequencing Logic , Control Unit Registers and Decoder, Control Memory
Multicore : CPU , Processor
↳ Core

* Cache Memory :
Multiple Layers of Memory, between the processor and main memory
speed up memory access

History of Computers

1) Vacuum Tubes → for digital logic elements and memory
   - ใช้สำหรับระบบปฏิบัติการ
   - IAS computer
   - CPU แบ่งเป็นส่วน Arithmetic-Logic Unit (ALU: CA) และ Program control Unit (CC)
   - IAS Memory Format → 40 bits สำหรับ op code + address และ each data type
   - Register MBR MAR IR IBR PC AC MQ
   - ทำงานเป็น 2 cycles หลักๆ → Fetch Cycle, Execution Cycle
2) Transistors
   - เล็กกว่า ถูกกว่า เย็นกว่า
   - high-level programming languages
   - ใช้ system software
3) Integrated Circuits (IC)
   - ส่วนประกอบเล็กลง ใช้ง่าย กินไฟน้อยลง
   - Moore's Law
     Number of transistors could be put on a single chip was doubling every year
4) Later generation
   - Semiconductor Memory Microprocessors

The evolution of the intel x86 Architecture

ต่อมาเป็น complex instruction set computers (CISCs)
และ reduced instruction set computer (RISC) ที่เล็กและประหยัด

Embedded system ex. เครื่องคิดเลข เครื่องซักผ้า

- IoT

4)

Cloud Computing

- Cloud Networking
- Cloud storage

② Performance Issues

Designing for Performance

- คุณภาพดีขึ้น ราคาถูกลงอย่างรวดเร็ว
  ↳ ความเร็ว

Microprocessor Speed

- Pipelining – แบ่งงานออกเป็นส่วนๆ แล้วส่งต่อกัน → 1 เส้นทาง → คล้ายโรงงาน
- Branch prediction – เดาว่าต้องกระโดดไปทำคำสั่งไหนต่อ
- Superscalar execution – มีหลายเส้นทางขนานกัน
- Data flow analysis – Processor analyzed which instructions + เตรียม hardware
- Speculative executation – ทำนายขนานรวมกัน

Performance Balance

key: ความเร็วสอดคล้องกัน

- Processor Component
- Main memory
- I/O Device
  → Interupting ♡

Improvements in Chip organization and architecture

- Increase hardware speed of processor
  gate เล็กลง → ใส่ gate ได้มากขึ้น → ใกล้กัน → increse clock rate
  ↳ กินไฟน้อยลง → ไวขึ้น
- Increase size and speed of caches
  cache access times drop
- Problem with clock speed and logic Density
  Power
  RC delay – resistance and capacitance
  Memory latency – access speed and transfer speed

Multicore, MICs, GPGPUs

Multicore – 2 cores or more on the same chip
caches รวม / caches แยก ?

MIC (many integrated core) – large number of core (>50 cores)
มีหลายงาน งานข้อมูลไม่เยอะ

GPU (graphic processing unit) – parallel operations on graphics data
1 งาน ข้อมูลเยอะ

Amdahl's and Little's Law

Amdahl's Law

$0 < f < 1$

เอางานทั้งหมดที่ใช้เวลา T พัฒนาส่วน fT (ไม่พัฒนาส่วน $(1-f)T$)
ให้เร็วขึ้น N เท่า → ทำงานบน N processor

♡ Speed up $= \frac{(1-f)T + fT}{(1-f)T + \frac{fT}{N}} = \frac{1}{(1-f) + \frac{f}{N}}$ $\frac{\text{rate ใหม่}}{\text{rate เก่า}} = \frac{\text{time เก่า}}{\text{time ใหม่}}$

∴ ทำ f เล็กๆ ปรับไปเท่าไหร่ก็ไม่ผล

Little's Law

♡ $L = \lambda W$

L items in queue
λ item arrive
W Wait Time

Basic Measure of Computer performance

Reduce Cycle Time with Little's Law → Increase TH + Reduce WIP

TH = throughput (arrival rate)
CT = cycle time (average time in system)
WIP = work in process (avg. number of units in system)

Processor performance Measure

- CPI = CPU cycle for program / Instruction Count (IC)
- Execution Time = wallclock = เวลารวม = I/O time + Program time
- MIPS = Millions of Instructions per second = $IC / (\text{Execution time} \times 10^6)$
- MFLOPS = Millions of Floating-Point Operations per Second

Calculating the mean

Normalized to A = $\frac{\text{Program in comp B}}{\text{program in comp A}}$

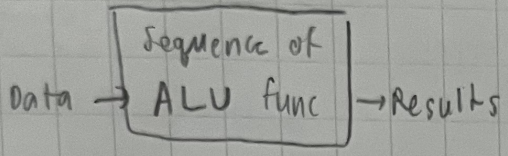
AM = $\Sigma x / n$
HM = $n / \Sigma \frac{1}{x}$
GM = $(\Pi x)^{\frac{1}{n}}$

Benchmark and SPEC

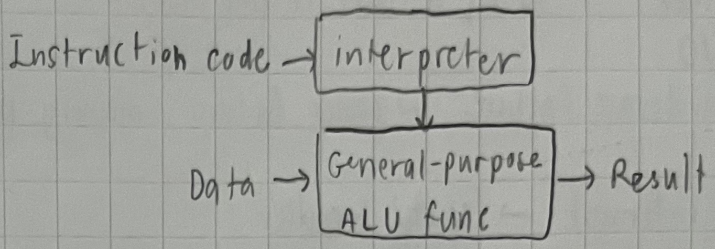
↳ โปรแกรมพิเศษ ๑๐ ข้อ ไว้เรียนใน comp sys arch

A top-level view of computer function and interconnection

Computer components



Programming in Hardware



Programming in software

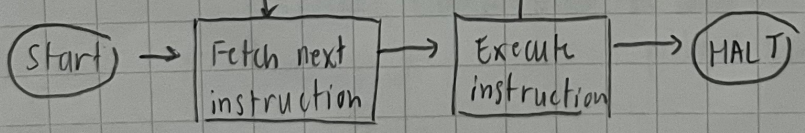
Software = instruction
= part of the hardware interprets each instruction and generate control signals

Major components
CPU - instruction interpreter
- module of general-purpose ALU funcs

I/O components
Memory address register (MAR) ...
Memory buffer register (MBR) ...
I/O Address register (I/O AR)
I/O Buffer register (I/O BR)
Program counter (PC)
Instruction register (IR)
Accumulator (AC) I/O for ALU

$$\frac{600(0.4)+400(0.6)}{\left(\frac{400}{3}\right)(0.6)+600(0.4)}$$

Instruction Fetch and Execute



- Fetch cycle
  - fetch instruction opcode + address
  - PC holds the address to be fetch next
  - PC is incremented after each instruction
  - fetched instruction is loaded into IR
  - processor interprets the instruction
- Execute cycle
  - processor - memory = data transfer btw CPU - main memory
  - processor I/O = CPU - I/O
  - Data processing = operation
  - Control = alternation of sequence of operation ~ Jump

Classes of Interupts

Program ~ arithmetic overflow, $\frac{0}{0}$, illegal machine instruction
Timer ~ by a timer in processor
I/O
Hardware failure ~ power failure, memory parity error

~ interupt → โปรแกรม handler
↳ เป็นแบบ nested

Interconnection structure

Module : Memory I/O CPU(processor)
Types of transfer
Memory → Processor (R)
Processor → Memory (W)
I/O → Processor (R)
Processor → I/O (W)
I/O ↔ Memory (R/W)

Bus Interconnection

- Data Bus
  no. of lines = width ∝ time
- Address Bus
  width ∝ maximum possible memory capacity
- Control Bus
  control the access and the use of data bus and address bus
  transmit both cmd and timing

Element of Bus Design

- Bus type : Dedicate (แต่ละอันมีของตัวเอง) / Multiplex (ใช้ร่วมกัน)
- Arbitration : Centralized / Distributed
- Timing : Synchronous / Async
- Bus width
- Data transfer type : Byte Transfer / Block data transfer

Point-to-Point Interconnect

การต่อแบบ Bus มีข้อเสียคือ
↳ บางอันเปลี่ยนไปต่อตรง ~ @PI : Quick Path Interconnect

peripheral Component Interconnect (PCI)

high bandwidth, hig speed I/O

Cache Memory

Memory System

Volatile / Nonvolatile (ต้องใช้ไฟเลี้ยง, ไม่ต้องใช้ไฟเลี้ยง)
Erasable / Nonerasable (ROM)

Memory Hierarchy

Speed ↓ Capacity

processor
- Inboard memory : Register Cache Mainmemory
- Outboard Storage : Magnetic Disk, CD-ROM, DVD
- Off-line storage : Magnetic Tape, MO, WORM

peripherals

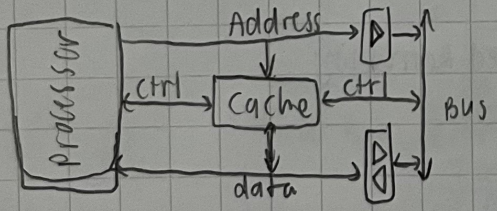
Key characteristics

- Location
- Capacity
- Unit of Transfer
- Access Method
- Performance
- Physical Type
- Physical Characteristics
- Organization

Access time = R/W time
Memory cycle time = เวลาของรอบ
$= \frac{1}{\text{transfer rate}}$

Cache memory principle

Small and fast, near CPU
include tags to identify which block of main memory is in each cache slot
word transfer with CPU, block transfer with main memory

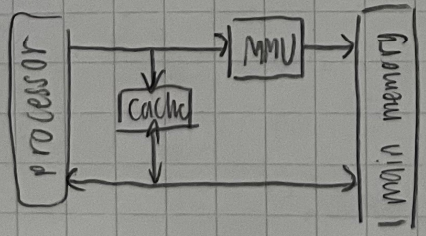


Elements of Cache Design

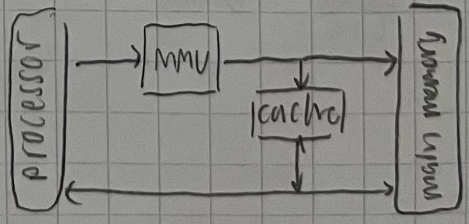
Cache addressing

MMU = memory management unit

Logical Address



Physical Address



mapping function

- Direct each block of main memory into only one possible cache line
- Associative each block can be loaded into any line of cache
- set Associative = 1+2

Direct Mapping

Memory Address format → Tag (s-r) + Line (r) + word (w)
↳ size of data (in bytes)

Associative mapping

Memory Address format → Tag + word
Compare ที่ทางวงจร
↳ Match tag ตรงกัน

ex Tag 16(00010110) Line 0CE7 (00110011100111) Word 00 Data FEDCBA98

↳ map to the same line in main memory
↳ ใน main memory จะเก็บได้สูงสุด 2^{14} แถว

Direct Mapping

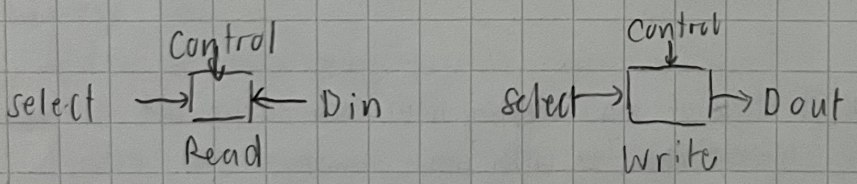
Set Associative Mapping → Tag + set + word
↳ Match Tag and set

5) Internal memory

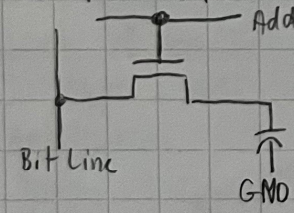
semiconductor main memory

RAM

- R/W, Volatile, Temp Storage, Static/Dynamic

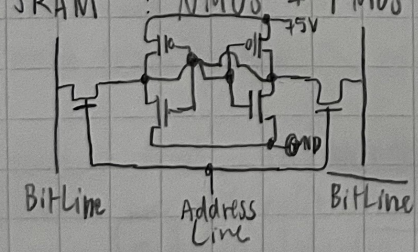


- DRAM : NMOS Transistor, Capacitor
  ↳ Leaking, Need Refreshing



  - Smaller per bit
  - Less expensive
  - Slower than SRAM

- SRAM : NMOS + PMOS + flip-flop



  - Faster
  - Larger per bit
  - on/off switch

ROM
- Permanent (non volatile)
- Microprogramming, Library Subroutines, BIOS (less change)
- Programmable ROM (PROM)
  - Erasable PROM (EPROM) → by UV
    - Electrically EPROM (EEPROM) → write กับ read
- Flash memory → Erase whole memory

Chip Logic
- Chip Select
- Refreshing
- MUX and Decoder

Interleaved Memory รวมหลายๆหน่วยความจำมาเป็นแบงก์ๆ access พร้อมกันได้ ~parallel
- a collection of DRAM grouped together → memory bank
- each bank is independantly R/W
- speed up

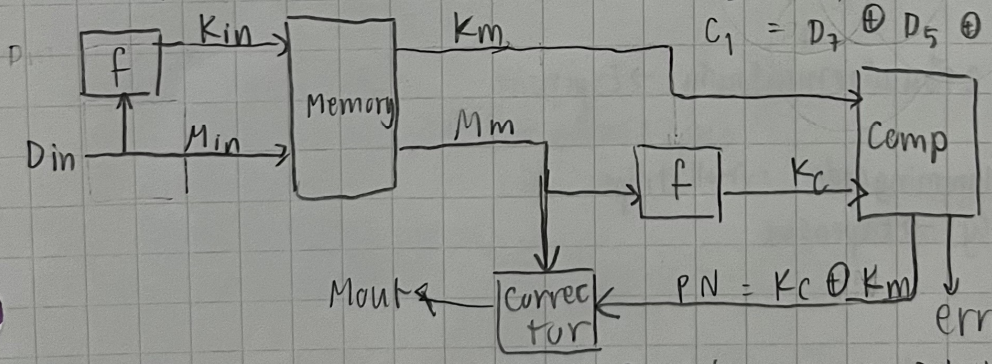
Error Correction

Hard Failure → permanent defect
Soft Error → random error

Error Correcting Code
ex → Hamming code
↳ using Venn Diagram

Error Correction Scheme
1) Error Detection : Parity Bit
2) Error Location : Error Syndrome
3) Error Correction

$C_8 = D_8 \oplus D_7 \oplus D_6 \oplus D_5$
$C_4 = D_8 \oplus D_4 \oplus D_3 \oplus D_2$
$C_2 = D_7 \oplus D_6 \oplus D_4 \oplus D_3 \oplus D_1$
$C_1 = D_7 \oplus D_5 \oplus D_4 \oplus D_2 \oplus D_1$



M databits + K check bits
$2^K - 1 \geq M + K$

Hamming SEC-DED Code (Single Correction, Double Detection)

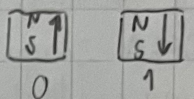
DDR DRAM

Synchronous DRAM
- RAM find data, CPU doesn't have to wait
- Ack signal
- Sent data twice per clock cycle
- ช่วงเวลาที่เสียไปตั้งแต่ได้รับสัญญาณขออ่าน ถึง เวลาที่อ่านได้จริง = latency

DDR SDRAM
- sent data once per clock

⑥ External memory

Magnetic Disk
- Aluminium coated with magnetizable material
- Glass
- มีหัวเขียน (แก้ค่า magnetic ใน cell) หัวอ่าน
- แบ่งข้อมูลเป็น sector (ขนาด 600 bytes) → winchester disk



RAID Redundant array of independent disk
- Not a hierachy

RAID 0
- No redundancy
- แบ่งข้อมูลเป็น stripe เก็บในดิสก์ที่ A % k → Round Robin striping
- Increase speed

RAID 1
- Mirrored Disk → มี 2 ก๊อปปี้ข้อมูลเหมือนกัน → Expensive

RAID 2
- Error correction : Hamming code - small stripe
  - Lots of redundancy → Expensive

RAID 3
- Pairity Bit
- high transfer rate

RAID 4
- Independently operate
- Large stripe
- Pairity disk

RAID 5
- pairity stripe across all disk
- Round robin pairity stripe

RAID 6
- 2 Parity Calculation
- 3 disks need to fail for data loss

⑦ Input / Output

- Slower than CPU/RAM
- Need I/O Modules

External device

- Human readable
- Machine readable
- Communication : Modem, Network Interface Card

I/O Modules

Function

- Control and Timing
- Processor Communication
- Device communication
- Data buffering
- Error detection

I/O Commands

- CPU issues address
- CPU issues command

Three techniques for I/O operation

1) Programmed I/O
2) Interupt driven
3) Direct Memory Access (DMA)

I/O Mapping

- Memory mapped I/O
  - Device & memory share an address space
- Isolated I/O
  - separate address spaces
  - Special command

Device Identification

- Multiple interrupt lines
- Software poll → ช้า
- Daisy chain
- Bus Master arbitration

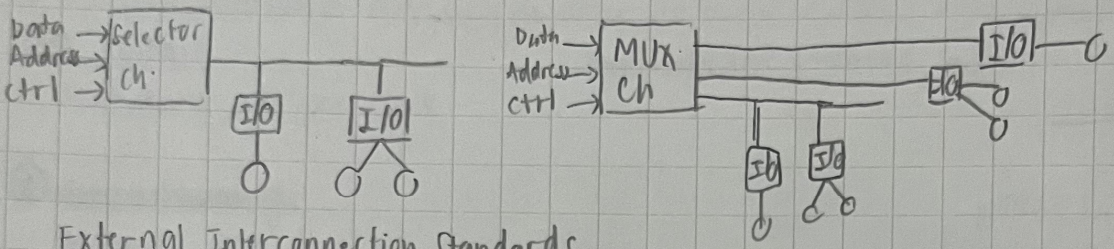
Priority decoder

DMA operation

- R/W, Device address,
  Starting address of Memory block for data
  Amount of data
- DMA ทำงานพร้อมๆ กับ CPU ได้เลย
- Sends interrupt when finished
- takes over bus / stealing
- CPU doesn't switch context

Direct Cache Access (DCA)
เพิ่มเติมจาก DMA

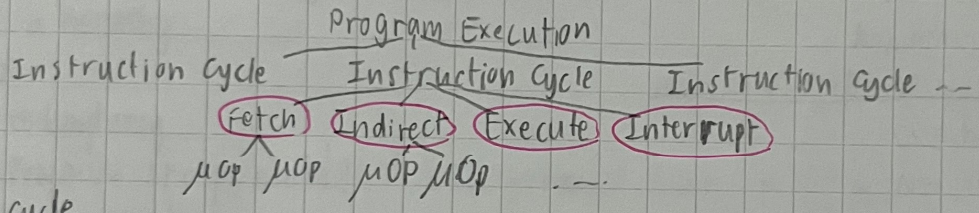
I/O Channels and Processors
I/O Channel Architecture



External Interconnection Standards
- Parallel I/O (to peripheral)
- Serial I/O (to peripheral, Parallel to serial)
- Point-to-Point / Multipoint

(20) Control Unit Operation

Micro-Operations



Fetch cycle

t_1 MAR ← (PC)
t_2 MBR ← (memory)
PC ← (PC)+1
t_3 IR ← (MBR)

or

t_1 MAR ← (PC) to bus
t_2 MBR ← (memory) from bus
t_3 PC ← (PC)+1
IR ← (MBR) free MBR

Rules for Grouping
- บางคำสั่งต้องเรียงลำดับการทำ เช่น ทำ MAR ← (PC) ก่อน MBR ← (memory)
- ห้าม R/W ใน Register เดียวกันพร้อมกัน
- บางอย่างทำ Parallel ไปพร้อมกับคำสั่งอื่นได้ เช่น PC ← (PC)+1 อาจใช้ ALU หรือ μOP ก่อน

Indirect Cycle
MAR ← (IR address)
MBR ← (memory)
IR address ← (MBR address)

Interrupt Cycle for saving data
t_1 MBR ← (PC)
t_2 MAR ← save-address
PC ← Routine-address
t_3 memory ← (MBR)

Execute Cycle ADD R_1, X

t_1 MAR ← (IR address) address of X

t_2 MBR ← (memory) value of X

t_3 R_1 ← (R_1) + (MBR)

Execute Cycle ISZ (increase and skip if zero)

t_1 MAR ← (IR address)

t_2 MBR ← (memory)

t_3 MBR ← (MBR) + 1

t_4 memory ← (MBR)

if (MBR) == 0 then PC ← (PC) + 1

Execute cycle BSA X (Branch and save address to X)

t_1 MAR ← (IR address)
MBR ← (PC)

t_2 PC ← (IR address)
memory ← (MBR)

t_3 PC ← (PC) + 1

Control of the processor

Functional Requirements

- Define basic elements of processor
- Describe μOP processor performs
  - Transfer data btw reg
  - Transfer data from reg to external
  - Transfer data from external to reg
  - Perform Arithmetic / Logical ops
- Determine funcs control unit must perform
  - sequencing
  - Execution

} use control signal

Control signals

- Clock
- IR - op-code
- Flags - state of CPU, results of prev op
- From control bus - Interrupts, Ack

(ex) MAR ← (PC) ต้องมี signal เพื่อเปิด gate ระหว่าง PC-MAR และปิด gate อื่นๆ ที่อาจเกิด conflict

Hardwire Implementation

- Control unit inputs
- Flags and control bus
- IR opcode - decoder
- Clock
- control unit outputs

problems
- ออกแบบได้ครั้งเดียว แก้ไขได้ยาก (Inflexible)
- ซับซ้อน

Micro-Programmed control

- Sequence of instructions to control complex operations

(14) Processor Structure and Function

Processor Organization

- Fetch instruction
- Interpret instruction
- Fetch data
- Process data
- Write data

เชื่อมต่อกันด้วย bus (internal / external)

Register organization

User-Visible Register user สามารถใช้งานได้

- General purpose - can be assigned to a variety of funcs
- Data - used only to hold data
- Address - ex. pointer
- condition code - flag

Control and Status Register Control unit ใช้ควบคุมการทำงาน

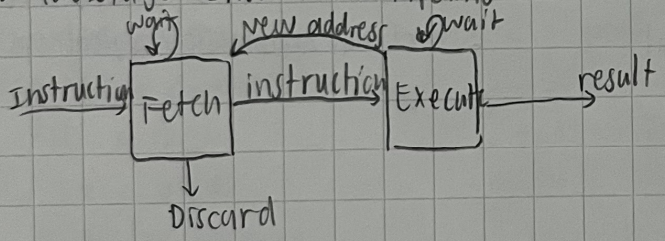
- PC
- IR
- MAR
- MBR

program status word (PSW)
sign zero carry Equal
Overflow Interrupt
Supervisor

Instruction pipelining

Pipelining strategy → ต่อคิว

- Two stage Instruction Pipeline



- Additional stage ⇝ FI DI CO FO EI WO

FI Fetch Instruction
FO Fetch Operands
DI Decode Instruction
EI Execute Instruction
CO Calculate Operands
WO Write Operand

- ⚠ Branch → Penalty → Priority

Pipeline Hazards

- Resource Hazard
  - Structural hazard
  - คำสั่งทำงานพร้อมกันเยอะคำสั่งมากไป เช่น fetch ซ้อนกัน → ต้องแบ่งใช้ memory กัน
  - เพิ่ม Idle state
- Data Hazard
  - เช่น บวกเลข แล้วเอาผลบวกมาใช้งานต่อ
  - เพิ่ม Idle state รอจนข้อมูลพร้อม หรือใช้ compiler แก้
  - Read after write (RAW) / true dependency
  - Write after read (WAR) / antidependency
  - Write after write (WAW) / output dependency
- Control Hazard / Branch Hazard
  - wrong decision
  - แก้โดยการ ทำทั้งคู่แล้วค่อยเลือก
    branch prediction
    delayed branch

(12) +13 Instruction Sets.

Machine Instruction Characteristic

Elements

- opcode
- source operand reference
- result operand reference
- next instruction reference

source operand reference, result operand reference] —
- Main / Virtual memory
- I/O device
- Processor register
- Immediate (constant after opcode)

Instruction Representation

- Opcode + operand reference + Operand reference
- ตามมาได้ 0-3 Address ต่อท้าย โค้ดสั้น คำสั่งยาว

Instruction set Design — ยังไม่มีรูปแบบที่ดีที่สุด

- Operation repertoire : จำนวน opcode ทั้งหมด
- Data types : รูปแบบข้อมูลที่รับ
- Instruction format
- Registers
- Addressing

Types of operands

Numbers : Binary int, floating point, BCD
Character : ASCII UNICODE EBCDIC
Logical Data : เช่นมาทำ and or xor กันได้ ของแต่ละ bit independently

Types of operations

Data transfer : Load Store Exchange(swap) Reset Set Push Pop | Location / Len / Mode
Arithmetic : + - x ÷ || negate ++ --
Logical : and or not xor comp shift rotate
Transfer of Ctrl : Jump Jumpif Return Exe Halt Wait
Input / Output :
Conversion : Translate Convert

Shift and Rotate ตัวอย่าง 10100110

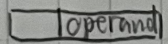
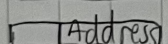
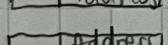
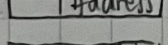
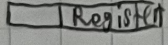
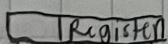
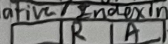
- Logical Right Shift 01010011 | Logical Left Shift 01001100
- Arithmetic Right Shift 11010011 | Arithmetic Left Shift 11001100 ต้องมี sign ใช้
- Right Rotate 01010011 | Left Rotate 01001101

Transfer of control

- Branch = กระโดด → condition / uncondition
- Skip = just increment PC
- Procedure Call Instruction → economy and modularity
  - ↳ call and return

ใช้ stack เก็บตำแหน่งที่จะกลับมาหลัง return ⇝ Stack frame

Addressing Mode

Immediate		
Direct		ใช้ค่าจาก Address ด้วยกันเอง
Indirect		ใช้ Address จาก Address ด้วยกันเอง (แล้วถึงค่าอีกที)
Register		ใช้ค่าจาก Register
Register indirect		ใช้ Address จาก Register (แล้วใช้ค่าจาก Address อีกที)
Displacement		ใช้ Address ใน Register + A = Address ใหม่ที่จะไปถึงค่า
Stack		ใช้ Top of stack register

Instruction format

Instruction length

- Memory size (to Access)
- Bus size
- Processor complexity and speed

fix length - มีที่ว่างที่ไม่ได้ใช้
flexible length - more complex

Allocation of bits

- Number of addressing mode
- Number of operand
- Address range.

(10) Computer Arithmetic

Integer Representation

- Sign-Magnitude
- Twos Complement
- Biased Representation (Bias=7) → บวกด้วยเลข Biased

Range Extension

- Sign-Magnitude ยาว bit เครื่องหมายไป bit ขวาสุด
- Twos Complement เลขบวกเติม 0 เลขลบเติม 1

Fixed-Point Representation = fixed radix point

Arithmetic - Negation Addition Subtraction Multiplication Division

Floating-Point Representation

scientific notation เพื่อให้มีความละเอียดมากขึ้น

32-bit floating point

signbit + 8-bits (biased-127) exponent + 23-bits significand

0 = +

ex $1.1010001 \times 2^{10100} = 0\ 10010011\ 10100010000\ldots0$

2 เว้นตำแหน่งไว้เลย

* มี underflow (ค่าที่ใกล้ 0 แต่ไม่ใช่ 0)

กฎพิเศษ

+0	= 0	00000000	000...0	$= 1.00 \times 2^{-127}$ } สองตัวน้อยเล็กมาก
-0	= 1	00000000	000...0	$= -1.00 \times 2^{-127}$
+∞	= 0	11111111	000...0	$= 1.00 \times 2^{128}$ } สองตัวใหญ่มาก
-∞	= 1	11111111	000...0	$= -1.00 \times 2^{128}$

การบวกลบ

- ต้องทำให้ exponent ของ 2 ตัวเท่ากัน แล้วค่อยบวกลบกัน
  - ปรับให้ละเอียดที่สุดก่อน ถ้า shift มากไป แปลว่ามีค่าก็ลืเศียง 0
- บวกกัน แล้วปรับ significant ให้อยู่ในรูป 1.xxx (shift left + increment exponent)

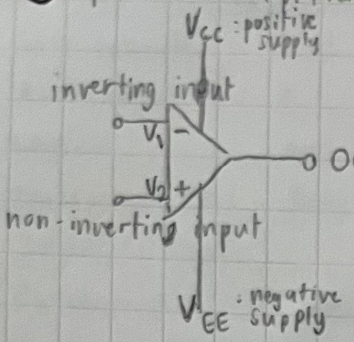
การคูณ

- bias subtraction in exponent
- เอา significant คูณกัน เอา exponent บวกกัน
- ปรับ significant
- bias addition in exponent

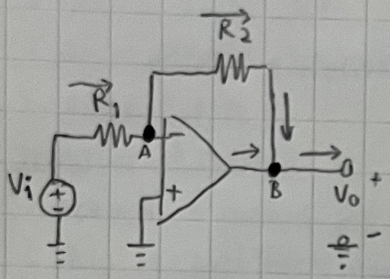
Guard bits

- มีตัวกั้น ตอนบวกลบกันจะได้ shift significant ได้ถูกต้อง
- ลด error

② Operational Amplifier (Op Amp) ตัวขยายสัญญาณ



- output = (Gain)$(V_2 - V_1)$
- Saturation
  output cannot exceed input
- Nonlinear trasfer
  ในช่วงที่ V_o เข้าใกล้ V_{CC}/V_{EE} เรียกว่า distortion
- Ideal opAmp
  $R_i = \infty$ $R_o = 0$ Gain $= \infty$



Inverting Closed-loop Configuration

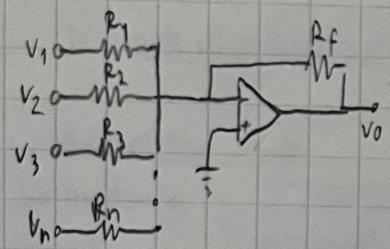
วิธีคำนวณ : Assume virtual ground

① Assume $V_1 = 0$

② ที่จุด A $V_A = 0 \rightarrow I_1 = \frac{V_i}{R_1} \rightarrow I_2 = I_1$

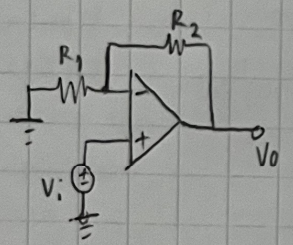
③ $V_o = V_B = -I_2 R_2 = -\left(\frac{R_2}{R_1}\right) V_i$

→ closed-loop gain



A weighted summer

$$V_o = -\left(\frac{R_f}{R_1} V_1 + \frac{R_f}{R_2} V_2 + \ldots + \frac{R_f}{R_n} V_n\right)$$



Non-inverting Configuration

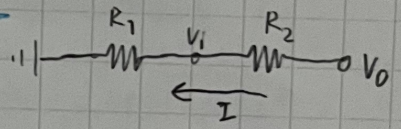
① $V_2 = V_i$

$V_2 - V_1 = 0 \rightarrow V_1 = V_i$

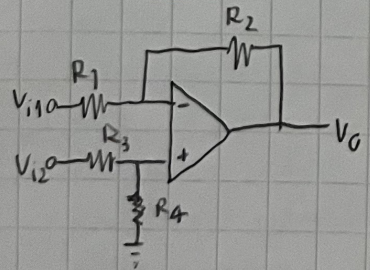
② $V_o = I(R_2 + R_1)$

$V_i = I(R_1)$

$V_o = V_i \left(\frac{R_2 + R_1}{R_1}\right)$ → Closed-loop gain



Difference Amplifier



ให้ $V_{i2} = 0$

$V_{o1} = -\left(\frac{R_2}{R_1}\right) V_{i1}$ A_1

ให้ $V_{i1} = 0$

$V_{o2} = \left(\frac{R_3}{R_3 + R_4}\right)\left(\frac{R_2 + R_1}{R_2}\right) V_{i2}$ A_2

$\therefore V_o = V_{o1} + V_{o2}$

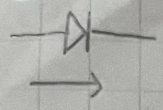
Superposition to Analysis

ให้ $V_{i1} = V_{i2} = V_{icm}$

แล้ว $V_o = 0$ (จาก $A_{cm} = 0$)

> unity gain buffer (gain = 1)
> = follower amp
> = buffer

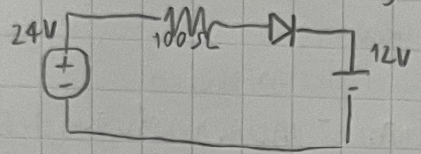
③ Diodes

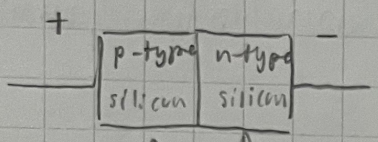


ยอมให้กระแสไหลทางเดียว
เป็น semiconductor

Basic knowledge
$V = IR$
Kirchoff's $i_{in} = i_{out}$

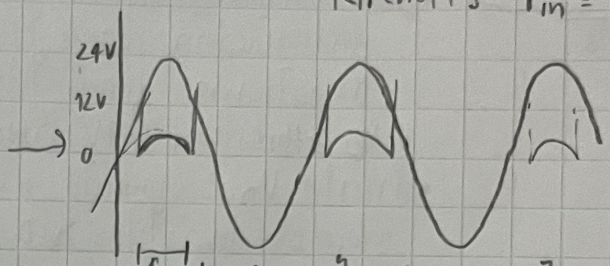
หลักการชาร์จไฟเข้า Battery



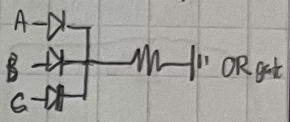


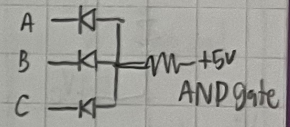
มีช่องว่าง ↑ ↑ มี 5 ขา

ex. replace B with Si ex. replace Si with P

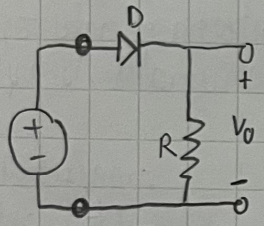


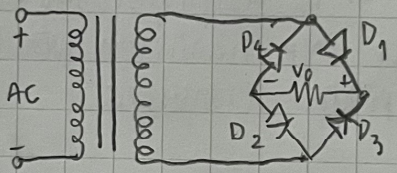
ช่วงที่กระแสไหลผ่าน diode ได้ = ชาร์จไฟ





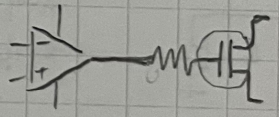
- แล้วจากนั้นโดน e^- ไปชักพา ขึ้นเกิด depletion region (free e^- รวมตัวกับ hole) มีผลต่อค่า Barrier voltage เกิดแรงต้านภายใน





ตัวอย่าง Op Amp

Comparator เหมือน OpAmp ทุกประการ โดย $V_+ > V_-$ = high
โดยปกติแล้วเราจะไม่ใช้ Comparator เพราะ ๆ เนื่องจากมี Noise มาก
เปลี่ยนไปใช้ Schmitt Trigger

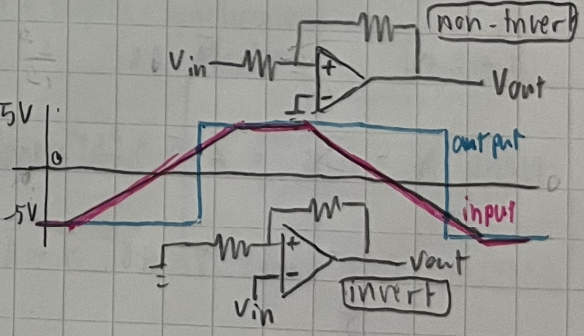


non-inverting

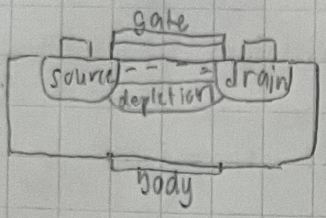
output lo→hi เมื่อ input lo→hi ผ่าน $\frac{R_1}{R_2} V_S$

hi→lo เมื่อ input hi→lo ผ่าน $-\frac{R_1}{R_2} V_S$

inverting เปลี่ยนจุดตัดเป็น $\pm \frac{R_1}{R_1+R_2} V_S$



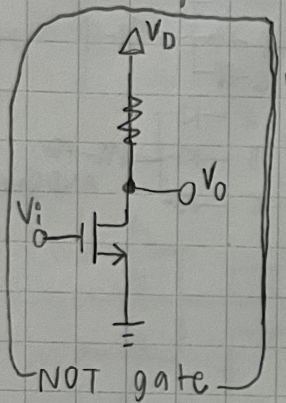
④ MOS Field-Effect Transistors (MOSFETs)



ให้เป็น NPN transistor

- ถ้าเราให้ศักย์สูงที่ด้าน gate จะดึงดูด minority carrier (e^- ใน body) ไปบนบริเวณที่ใกล้ gate เกิด depletion region
- เมื่อเราใส่ศักย์สูงที่ด้าน drain จะทำให้
  - electron ไหลจาก source ไป drain
  - กระแสไหลจาก drain ไป source
  - $V_G \geq$ threshold $\approx 1V$ ถึงจะเริ่มมี depletion region
- ถ้าเราใส่ V_D มากเกินไป จะทำให้แรง e^- แรงลงและต้านลงที่ฝั่ง D เพราะ e^- โดนดึงไปอยู่ใน D มากเพิ่มมากเกินไปจะทำให้ถึงจุดอิ่มตัว

- สำหรับ PNP transistor ให้สลับ V_G และ V_D
- ส่วนใหญ่จะต่อ S กับ B เข้าด้วยกัน



$V_i = V_G \quad V_o = V_D$

ขณะที่ $V_i < V_T \rightarrow V_o = V_D$

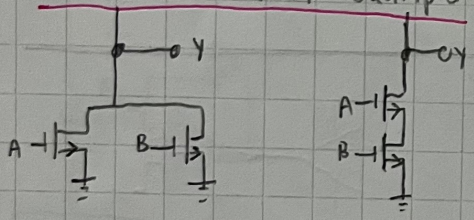
$0 \leq V_i - V_T \leq V_D \rightarrow V_o \approx V_D - V_i$

else $\rightarrow V_o$ ดูเข้าจุดหนึ่งที่ใกล้ 0 (ไม่เป็น 0 เนื่องจาก R ภายใน T)

ก็เลยเปลี่ยน R เป็น PNP Transistor เพื่อให้วงจรต่อกับขาดได้อย่างสมบูรณ์

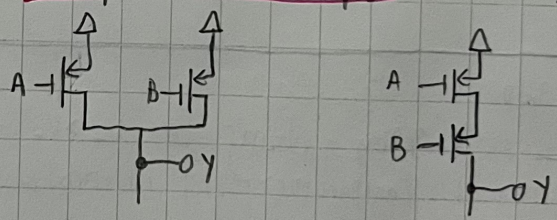
เรียก Pullup - Pulldown transistor

Pull-down network example



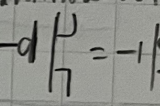
$\bar{Y} = A + B$ $\qquad \bar{Y} = AB$

Pull-up network example



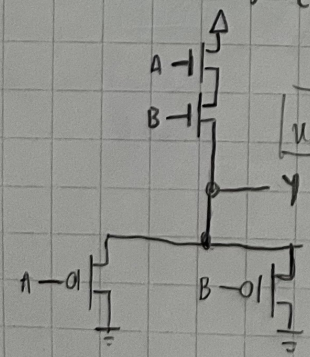
$Y = \bar{A} + \bar{B}$ $\qquad Y = \bar{A}\bar{B}$

การสลับ bellike:



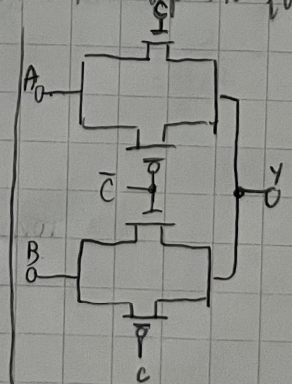
NAND $Y = \overline{A \cdot B} = \bar{A} + \bar{B}$ → เอารูปซ้ายของ Pull-up ต่อ รูปขวาของ Pull-down

exercise AND gate $Y = AB$, $\bar{Y} = \bar{A} + \bar{B}$



หลักเลี่ยง

การทำ NAND+NOT เนื่องจากปัญหาทางไฟฟ้า ทำให้ $V_G \not> V_T$



MUX 2:1

Pass-transistor

① Embedded system Communication protocols

Communication protocols

- Asynchronous (no shared clock) / synchronous (Master has a clock)
- Serial / Parallel
- Intersystem (between 2 devices) / Intra system (internally)
- Mode of communication
  - Simple — one direction
  - Half duplex — one direction at a time
  - Duplex

★ UART : Universal Asynchronous Receiver / Transmitter

- Translate parallel ↔ serial
- used in conjunction (Physical Level)
- Each char is sent as

  start bit → number of data bits → parity bit (เช็คความถูกต้อง) → stopbit → baud rate (bit timing)

  (ex.) 9600-N-8-1 → baudrate parity no.of data stop bit ↳ หน่วย bit/second (Hz)
- Signals

  RXD Receiving The data
  TXD Transmitting the data
  } ต่อไขว้กัน

  optional { RTS# Ready to send
  CTS# clear to send
  } ต่อไขว้กัน

  (ex.) การส่ง ตัว J (Hex 4A → 0100 1010)

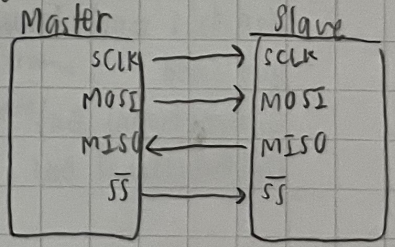
  waiting → 1 | 0 0 1 0 1 0 0 1 0 | 1 1 ← waiting
  start → ส่งเรียงจาก bit ต่ำไป bit สูง → stop

  ∴ ส่ง 1 ตัวอักษร ใช้ 10 bits
- UART speed depens on Electrical & Clock

★ SPI : Serial peripheral Interface → serial + Duplex + Synchronous

- Signals

  SCLK Clock
  MOSI Master out - Slave In → ต่อไขว้ตรงกัน ไปไขว้
  MISO Master In - Slave Out
  nSS/CS NOT slave selected



- อ่านข้อมูลเร็วกว่า UART
- ใช้สายไฟเยอะกว่า UART
- ถ้าไม่ส่งสัญญาณ SS สัญญาณ MISO จะเป็น Z → ทำหน้าที่เป็น OE
  → เปลี่ยนไปต่อแบบ Daisy-Chained (ต่อ MISO ของตัวแรก ไป MOSI ของตัวถัดไป)
- ทำงานแบบ shift register เอาค่าไปใช้งานได้ตอนที่ ชิพไม่ทำงาน เฉพาะที่ risingedge
- ส่งจาก Lower bit → higher bit
- เน้นไว้ที่ ตัวแปลงสัญญาณ digital to analog / analog to digital

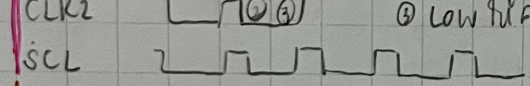
★ อันที่มี diagram shift อย่า

☆ I^2C : Inter-integrated circuit (I squared C) → Synchronous + Half Duplex

- นิยมใช้ใน intracommunication ex. สาย HDMI
- เป็น Bus Protocol
- Multi Master / Slave
- ทำงานบนสายไฟ 2 เส้น → SDA (data) / SCL (clock) สั่งได้จาก master เท่านั้น
- มี Address ส่วนใหญ่เป็นแบบ 7 bit
- Multi Master with Collision Detection (ค่อยๆดูทีละ bit) ทำได้เพราะมี clock sync + data arb.
- ค่อนข้างช้า (ประมาณเท่า UART)
- Wired-AND ทำให้ไม่ชัดๆ
- Conditions and Data Validity
  - SDA falling = initialize
  - SCL Low = first data change
    High = เก็บ data
  - SDA rising = end

  } ตรงข้ามกับ SPI
- ส่งข้อมูลอย่างน้อย 2 bytes
- ถ้าส่งสัญญาณถูก จะมีสัญญาณ Ack
- ส่งแบบ 8 bit แรก 7 bit แรก = address 1 bit สุดท้าย = R/W
  ส่งสัญญาณ "พร้อมแล้ว" กลับมา
  เขียน/อ่าน ยังไม่จบ จนกว่าจะมีสัญญาณ "พอแล้ว" กลับมา

• Clock synchronization



① ควรจะ high แต่ไม่ high เพราะตัด CLK 2 = high wait state
② ควรจะ high แต่ไม่ high เพราะตัด CLK 1 } จนไป
③ Low ให้ครบคาบของตัวเอง ④ เลย high ไม่ได้ }
← ช่วงขาขึ้น 50% ก็ได้

• Data Arbitration

SDA เกิดจากการทำ Data1 AND Data2
ถ้าอ่านข้อมูลแล้วตรงกันก็ไป แต่ถ้าส่งตัวใดตัวหนึ่งพยายามส่งข้อมูล 1 ในขณะที่อีกตัวส่ง 0
ตัวที่ส่ง 1 จะแพ้แล้วยอมแพ้ไป (loses arbitration)

- Bus Speed
  Bidirectional Bus (Standard Mode / Fast Mode / Fast Mode Plus / High Speed Mode)
  Unidirectional Bus (Ultra Fast Mode)

☆ CAN : Controller Area Network → Asynchronous + Half Duplex + Multi Master/Slave

- Complex - พบได้ในรถยนต์
- ต่อ Microcontroller กับ CAN controller และ CAN transceiver
- Prioritization of Message
- Guarantee of Latency Times
- มีสายไฟ 2 เส้น CANH, CANL
- Data Consistency (ทุกคนต้องอยู่แบบเดียวกัน)
- Error Detection + Error Signaling
  สามารถ handle temp error / permanent error ต่างกันได้ (เลิกคุยกับตัวนึงได้เอง)

- Frame Formats (11 bits / 29 bits) ≈ address
- Frame Type (Data Frame / Remote frame / Error Frame / overload Frame)
- Frame Fields ↳ ข้อมูลจริงๆ ↳ การขอข้อมูล
- Idle state เป็น 1
- SOF (start of frame) = 0
- Arbitration field มี 11-bit Identifier มีแบบ 12 bits และ 32 bits
  มี RTR หรือ SRR (บอกว่า มี Identifer อีก 18 bits มั้ย)
  ↳ ถ้ามี SRR ตรับ Identifier อีก 18 bits แล้วรับ RTR
- ต่างกัน • Data Frame ประกอบ control, data (ไม่เกิน 8 bytes), CRC, ACK, และ End of Frame / Error Frame / Overload Frame
  • Remote Frame ไม่มี data, RTR=recessive ↳ เช็คว่าข้อมูลที่ส่งถูกต้องมั้ย
  • Error Frame ต่างจาก Data Frame / Remote Frame ↳ ไม่ขึ้นกับว่า Low/High
    มี 2 fields → Error Flag, Error Delimiter → 8 recessive bits
    ↳ Active error flag 6 consecutive dominant bits
    ↳ Passive error flag 6 consecutive recessive bits
  • Overload Frame ต่างจาก EOF / Error Delimiter / Overload delimiter
    มี 2 fields → overload flag, overload delimiter
    ↳ 6 dominant bits ↳ 8 recessive bits
    มี 3 เงื่อนไข → require a delay
    → detect a dominant bit during intermission
    → detect a dominant bit at last bit
- ช่อง Data ไม่เกิน 8 bytes ทำไม่ใช้เยอะ ตรวจสอบการเกิด error
- CRC Field ประกอบด้วย 15 bits CRC sequence + 1 recessive bit CRC delimiter
- Ack Field ประกอบด้วย 1 ack bit (ส่ง/ไม่ส่ง) + 1 recessive bit ACK delimiter
- Fault Confinement → use counter

Data Frame
start Arbitration Control Data CRC ACK End
Overload frame
Error frame

Remote Frame
Start Arbitration Control CRC ACK End
Overload frame

Error Frame → Error flag Error Delimiter + Next frame / Overload frame

Overload frame → Overload flag + overload Delimiter + Next frame / overload frame

UART	Async			
SPI	Sync	Serial	Duplex	Multislave
I^2C	Sync		H-Duplex	Multimaster/Slave
CAN	Async		H-Duplex	Multimaster/Slave

☆ USB : Universal Serial Bus → Async Half Duplex (USB 2.0) + Master/Slave

- Upstream Connection and Downstream connection
- Device Speeds (ความเร็วในการรับส่งสัญญาณ) มีได้ 4 รูปแบบ
- Types of Connectors (A/B/C) and Sizes (Standard/Mini/Micro)
- Interface
  VCC / D⁻ / D⁺ / GND / SSTX± / SSRX±
  (SSTX± / SSRX± : USB 3.0 3.1 gen2 มี 2 คู่)
- Architecture : Tiered star Topology → Host Controller + Hubs + peripherals
  Host : Enumeration + Learn Identity (VID / PID)
  Hub : increase the logical and physical fan out (Bus Powered Hub / Self Powered Hub)
  Peripherals : Receive and respond to the command (Standalone / Compound Device)
- Power Management : enable / disable power to the device
  host / hub คิดได้ว่าไฟเข้า/ไม่เข้า
- ส่งข้อมูลได้ 4 แบบ
  Control Transfer config cmd
  Bulk Transfer ส่งข้อมูลเยอะ แต่ไม่รีบ ~ printer
  Isochronous Transfer ~ webcam ethernet ต้องส่งให้เร็วประมาณนึง
  Interupt Transfer ถาม slave ว่า คุณมีข้อมูลให้ผมมั้ย → เปลืองไฟ
- OTG : On the Go เป็นได้ทั้ง Host และ Peripheral
- End point = unique point, definite address (4 bit)
  แต่ละ device มี 16 end points, end point 0 สำหรับ control transfer
- Host ต่อกับ peripheral ผ่าน Virtual pipe
- เมื่อต่อกันได้ peripheral จะส่ง Descriptor (Data struct about config) ไปให้ host

② Control and Feedback

Control = basic function , involves regulation or cmd.

Open-loop systems
- Simple control
- user has a goal , select input , get some output (ที่อาจจะไม่ตรงกับ goal)

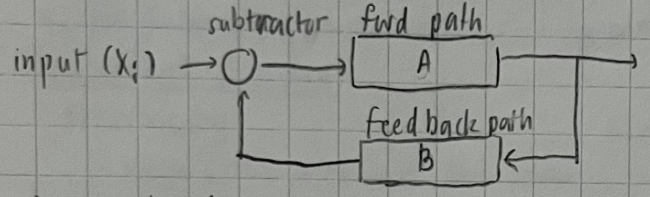
Closed-loop systems
- user inputs the goal to the system
- มี comparator เพื่อปรับแก้ไขค่า

Automatic Control systems

ex. อุณหภูมิห้อง

Input (เป้าเครื่องปรับ) → Fwd path → Heater/A/C → output (Room Temp)
Temp Sensor ← Feedback path ←

Feedback Systems



ที่ subtractor $X_i - \frac{X_o}{A} = BX_o$

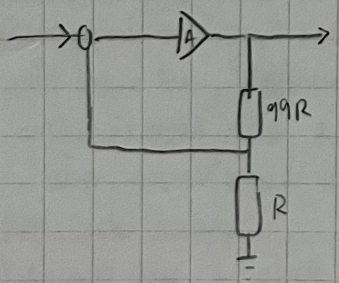
Assume ว่าเป็นอนุกรูอน

Gain $G = \frac{X_o}{X_i} = \frac{A}{1+AB}$ ← transfer function

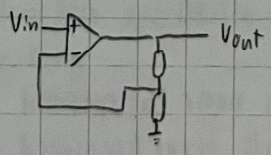
$AB < 0$ = positive feedback
$AB > 0$ = negative feedback ที่ A เยอะมากๆ $G = \frac{1}{B}$

Negative Feedback
- variability (การเปลี่ยนแปลง)

สมมติว่าเรารู้แล้วว่าเราอยากได้ $G = \frac{1}{B} = 100 \rightarrow B = 0.01$
เราสามารถต่อวงจรได้ ดังนี้



Operational amplifier (op-amp)
≈ open-loop ที่โดนจำกัดแค่ G แต่ห้ามเกิน $\frac{1}{B}$
→ กลายเป็น closed-loop



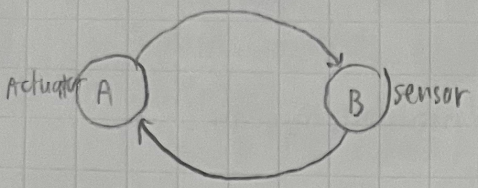
Effect
- reduce gain $1+AB$ เท่า
- frequency response (increase bandwidth)
- distortion and noise ลดลง

Property
- independent output
- overall is determined by feedback path

→ feedback เพิ่ม stability สูง

③ PID Control for Embedded System

Feedback Control



- PID = Proportional, Integral, and Derivative Contrd เป็น 1 ในรูปแบบของ feedback control
- ถ้า feedback control (simple) เป็นแบบลูกโลก → ไม่ smooth เพราะมี inertia

- Proportional Control

  error = setpoint - current

  output = P = Kp × error ; Kp = Proportional Gain

  - ถ้า Kp ใหญ่เกินไป ก็จะกลับไปแกว่งแบบแรก
  - ถ้า Kp เล็กเกินไป output จะค่อยๆเข้าหาค่าเมื่อน้อยกว่า setpoint

- Proportional-Integral Control

  accumError = sum of error

  I = Ki × error ; Ki = Integral Gain

  output = P+I

  - ทำให้กราฟพุ่งเข้า setpoint ได้ไวกว่า I

- Proportional - Derivative control

  errorDiff = error - lasterror

  D = Kd × errorDiff ; Kd = Derivative gain ; D = dampening → add stability

  output = P+D

- Proportional - Integral - Derivative Control

  output = P+I+D

Saturation - Common Mistake

- output อาจเป็นค่าไหนก็ได้ แต่อุปกรณ์รับค่าได้แค่ช่วง [A,B]
- ใช้ saturate (if <min → min if >max → max)

Timesteps - Common Mistake

- ต้องใช้ timesteps ที่ uniform

  ไม่งั้นแก้ด้วย Ki = Ki × dt;

  Kd = Kd / dt;

PID Tuning

Tune P : full power unless near the setpoint

Tune Ki : until steady-state error is removed

Tune Kd : dampen overshoot, improve responsiveness

Interrupt - DMA

I/O Types

- **Programmed I/O** (Polling / Handshaking I/O)
  - เช็คตลอดเวลาว่ามี I/O ในฝั่งเรียบ, direct control, เสียเวลา
  - ทำงาน 3 อย่าง
    CPU Request
    I/O Module performs operation
    CPU checks status bits periodically
  - ทำงานบน address
- **Interrupted Driven I/O** executing ISR
  - ส่งไปทำ+ตัว แล้วกลับมาบอกว่า ส่งเสร็จแล้ว จะส่งอันต่อไปแล้วไหม
  - CPU ใช้เสร็จจะทำงานไปทำอย่างอื่นได้ ไม่ต้องคอยเช็ค
  - Interrupt ไม่มีการส่งข้อมูล เป็นการบอกว่า มีอะไรเกิดขึ้น ให้ CPU ไปดูแปลว่า
  - มีการ push stack และเปลี่ยนแปลง PC
  - เช็ค Interrupt ทุกการจบ instruction cycle

Interrupt มี 2 ส่วน

1) servicing
   - push stack + change stack pointer
   - change program counter
   - ทำงานไปจนเสร็จ และเน้นว่าต้อง return
2) return
   - เอาข้อมูลใน control stack ใส่กลับไปที่เดิม
     - Program Counter
     - Register

ปัญหา: - จะรู้ได้ไงว่า module ไหนเป็นคน interrupt
- จัดการ handle หลาย interrupt → priority / Bus Master

วิธีแก้
1) **แยกสาย** interrupt ทั้งหมด ง่าย แต่ limit จำนวนสายไฟ
2) **software poll** ถามทุกคนเรียงกัน เสียเวลา
3) **Daisy Chain** / Hardware poll มีปัญหาเรื่อง priority
4) **Bus Master** ใครขอ interrupt ก่อน จะได้เป็น Bus Master

- **Direct Memory Access (DMA)**
  - เกิดจากการที่ การสั่งพิมพ์ แล้วรอเฉยๆ ทำให้เกิด interrupt หลายครั้งมาก ช้ามากๆ
  - DMA อยู่บน Bus, ไม่อยู่บน CPU
    - CPU สั่ง DMA ทำให้ R/W จาก address ไหน ของ Device Address ไหน และความยาวของข้อมูล
    - DMA ทำงานเสร็จ จะส่ง interrupt / polling

DMA Cycle Stealing

- Take over bus 1+ clock cycles
- 1 การ Steal จะส่ง 1 word
- ทำให้ CPU ช้าลง แต่ก็ยังเร็วกว่า interrupt

Config

1) ทุกอย่างอยู่บน Bus 1 transfer ต้องแย่ง CPU 2 ครั้ง
2) hierachy แต่ละ I/O device ก็มี DMA controller 1 transfer แย่ง CPU 1 ครั้ง
   - ไม่สามารถ Access 2 devices ที่อยู่ใต้ DMA เดียวกันพร้อมกันได้
   - Device คุยกันเองโดยไม่ผ่าน CPU ไม่ได้

3) มี I10 Bus เป็นของตัวเอง transfer ทุก CPU เวลาใช้

I-Bus : Instruction
D-Bus : Data
S-Bus : Bus ที่ไม่มี cache

Example :
ARM and STM32F4XX Operating modes & Interrupt Handling

- มี NVIC (Nested vectored Interrupt Controller) เป็นตัวจัดการรับและ prioritize Interrupt ทั้งหมด
  ข้อดี : ตัวแปรใน processor จะทำได้เลย เพราะใช้ compiler เดิมได้
- สำหรับ Stack Pointer มี 2 อัน
  Process SP สำหรับ handler / thread mode
  Main SP สำหรับอื่นๆ
- Exception States
  Inactive
  Pending
  Active
  Active and Pending
- Tail chaining Interrupts
  . ประหยัดเวลาการ push-pop stack

Booting Process

- กด Reset แล้วจะทำ Non-maskable interrupt (cannot ignore)
- มี 2 Register ที่ชี้ตำแหน่งอยู่ที่ address 0 คือ top of stack และ Reset Handler
  • Reset handler ทำงานแก้ไขค่าใน Global vars ให้เป็นไปตามที่ควร แล้วเรียก main()
- Boot Mode

Boot1	Boot0	Mode
x	0	from main flash memory
0	1	from system memory (bootloader)
1	1	from embedded SRAM

Boot จากอุปกรณ์อื่นได้ } to Alias ที่ address 0

- อาจมี Boot loader sequence เพราะ RAM ที่อยู่ไม่พอ