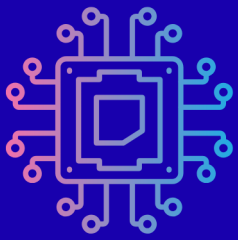
2026 Edition

The Future Coder Series

Computer Science စာအုပ်

Vol 1: Computer Systems



ရန်နိုင်လင်း (Software Engineer)

MBA, PG Dip (CS), BSc(Physics)

Chapter 1: The Modern Computer System

Chapter 2: Data Representation

Chapter 3: Information and Logic

Chapter 4: Hardware

Chapter 5: Data Structures & Algorithms

Chapter 6: Memory Organization

Chapter 7: System Software & Operating Systems

7.4 Client-Server နှင့် Cloud Computing

Chapter 8: Programming Languages

8.1 Programming Language အဆင့်ဆင့်

8.2 Language Processors (programming language တစ်ခုနဲ့တစ်ခုပြောင်းပေးတဲ့ program)

8.3 Programming Standards (စံနစ်တကျ ရေးသားခြင်း)

Chapter 9: Network Technology

9.1 Network ဆိုတာ ဘာလဲ?

9.2 Network Topologies (ကွန်ရက် ချိတ်ဆက်ပုံစံများ)

9.3 Transmission Media (ချိတ်ဆက်သည့် နည်းလမ်းများ)

9.4 The Internet & Connection (အင်တာနက် အလုပ်လုပ်ပုံ)

Chapter 1: The Modern Computer System

(ကွန်ပျူတာစနစ်နှင့် ခေတ်သစ်နည်းပညာများ)

1.1 ကွန်ပျူတာဆိုတာ ဘာလဲ? (What is a Computer?)

လူအများစုက ကွန်ပျူတာဆိုရင် စားပွဲတင်စက်
(Desktop) သို့မဟုတ် Laptop ကိုပဲ ပြေးမြင်တတ်ကြပါတယ်။
ဒါပေမဲ့ Computer Science သဘောတရားအရ ကွန်ပျူတာဆို
တာ **"အချက်အလက်တွေကို ယူမယ်၊ အလုပ်လုပ်မယ်၊ ရလဒ်ပြန်ပေးမယ်"**
(Input–Process–Output) ဆိုတဲ့ အလုပ် ၃ ခုကို လုပ်နိုင်တဲ့ စက်ပစ္စည်းတိုင်း
ကို ခေါ်ပါတယ် ။
ဒါကြောင့် လက်ထဲက **Smart Phone**၊ လက်ပတ်နာရီ **Smart Watch** နဲ့ အိမ်
က **Smart TV** တွေအားလုံးဟာ ကွန်ပျူတာတွေ ဖြစ်ကြပါတယ်။
လွယ်ကူတဲ့ ဥပမာ (ထမင်းပေါင်းအိုး တစ်လုံးကို မြင်ကြည့်ပါ) –

1. **Input:** ဆန်နဲ့ ရေ ထည့်လိုက်တယ် (အချက်အလက် ထည့်သွင်းခြင်း)။
2. **Process:** အိုးက အပူပေးပြီး ချက်ပြုတ်လိုက်တယ် (လုပ်ဆောင်ခြင်း)။
3. **Output:** ထမင်းဖြစ်သွားတယ် (ရလဒ်ထွက်လာခြင်း)။ ကွန်ပျူတာက
   လည်း ဒီသဘောတရားအတိုင်း အလုပ်လုပ်ပါတယ်။

1.2 ကွန်ပျူတာရဲ့ အဓိက အစိတ်အပိုင်း (၂) ခု

ကွန်ပျူတာတစ်လုံး ဖြစ်လာဖို့ **Hardware** (ရုပ်) နဲ့ **Software** (နာမ်) နှစ်
ခု ပေါင်းစပ်ရပါမယ် ။

(a) Hardware (ကိုင်တွယ်ထိတွေ့၍ရသော အပိုင်း)

မျက်စိနဲ့မြင်ရပြီး လက်နဲ့ကိုင်လို့ရတဲ့ အစိတ်အပိုင်းတွေ ဖြစ်ပါတယ်။ ဒါကို **လူ ခန္ဓာကိုယ်** နဲ့ နှိုင်းယှဉ်ကြည့်ရအောင် ။

1. **Input Devices (အာရုံခံ အစိတ်အပိုင်းများ):**

- ကွန်ပျူတာထဲကို အချက်အလက်ပို့ပေးတဲ့ အရာတွေပါ။
- လူမှာဆိုရင်: မျက်စိ၊ နား၊ နှာခေါင်း။
- ကွန်ပျူတာမှာဆိုရင်: **Touchscreen** (တို့ထိတာကို သိတယ်)၊ **Mic** (အသံကို နားထောင်တယ်)၊ **Camera** (ပုံရိုပ်ကို မြင်တယ်)၊ **GPS** (ရောက်နေတဲ့ နေရာကို သိတယ်) ။

2. **Central Processing Unit – CPU (ဦးနှောက်):**

- ဒါက အရေးကြီးဆုံး အစိတ်အပိုင်းပါ။ ဝင်လာတဲ့ အချက်အလက်တွေကို စဉ်းစားတွက်ချက်ပေးပါတယ်။
- လူမှာဆိုရင်: ဦးနှောက်။
- ကွန်ပျူတာမှာဆိုရင်: ဖုန်းထဲက **Chipset** (Snapdragon, Apple A-series) သို့မဟုတ် ကွန်ပျူတာထဲက **Intel/AMD Processor** တွေ ဖြစ်ပါတယ် ။

3. **Memory & Storage (မှတ်ဉာဏ်):**

- **RAM (ယာယီမှတ်ဉာဏ်):** စာမေးပွဲခန်းထဲမှာ ခဏတာ အလွတ်ကျက်ထားသလိုပါပဲ။ စက်ပိတ်လိုက်ရင် ပျောက်သွားပါတယ်။ အလုပ်လုပ်နေတုန်း ခဏသုံးတဲ့နေရာပဲ။
- **Storage (ရေရှည်မှတ်ဉာဏ်):** ဒိုင်ယာရီစာအုပ်ထဲ ရေးမှတ်ထားသလိုပါ။ ဖုန်းပိတ်လိုက်လည်း ပျောက်မသွားပါဘူး။ ဓာတ်ပုံတွေ၊ သီချင်းတွေ သိမ်းတဲ့နေရာ (128GB, 256GB) ဖြစ်ပါတယ် ။

4. **Output Devices (တို့ပြန်ပြသသော အစိတ်အပိုင်းများ):**

- တွက်ချက်ပြီးသား ရလဒ်ကို လူတွေနားလည်အောင် ပြန်ပြပေးတဲ့အရာပါ။

- လူမှာဆိုရင်: ပါးစပ် (စကားပြောဖို့)၊ လက် (စာရေးဖို့)။
- ကွန်ပျူတာမှာဆိုရင်: **Screen/Display** (ရုပ်ပြဖို့)၊ **Speaker** (အသံကြားရဖို့) ။

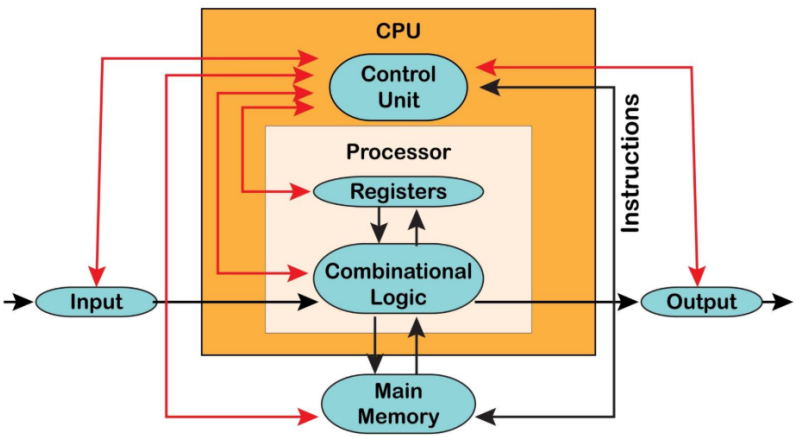


image - Shutterstock

(b) Software (ညွှန်ကြားချက်များ)

Hardware ကြီးပဲရှိပြီး Software မရှိရင် ကွန်ပျူတာက သံတိုသံစပါပဲ။ Software ဆိုတာ "Hardware ကို ဘာလုပ်ရမယ်ဆိုတာ ခိုင်းစေတဲ့ ညွှန်ကြားချက်များ" ဖြစ်ပါတယ် ။

1. **System Software (လည်ပတ်ရေး စနစ်):**
- စက်တစ်လုံးကို စဖွင့်လိုက်တာနဲ့ အလုပ်လုပ်တဲ့ စနစ်ပါ။
- ဥပမာ - ဖုန်းမှာဆို **Android** သို့မဟုတ် **iOS**၊ ကွန်ပျူတာမှာ ဆို **Windows** ပါ။ သူတို့မရှိရင် ဖုန်းခေါ်လို့မရ၊ ဂိမ်းဆော့လို့မရပါဘူး ။
2. **Application Software (အသုံးချ ဆော့ဖ်ဝဲ):**
- ကိုယ်သုံးချင်တဲ့ ကိစ္စတစ်ခုချင်းစီအတွက် ထည့်ထားတာပါ။
- ဥပမာ - **Facebook** (လူမှုရေး)၊ **Mobile Legends** (ဂိမ်း)၊ **CapCut** (ဗီဒီယို တည်းဖြတ်) စတာတွေ ဖြစ်ပါတယ် ။

1.3 ကွန်ပျူတာ အမျိုးအစားများ (Types of Computers)

အရင်ခေတ်က ကွန်ပျူတာတွေကို အရွယ်အစားနဲ့ ခွဲခြားခဲ့ပေမယ့် ၊ ဒီနေ့ခေတ်မှာတော့ စွမ်းဆောင်ရည်နဲ့ အသုံးပြုပုံအပေါ်မူတည်ပြီး ဒီလိုခွဲခြားနိုင်ပါတယ် -

1. **Personal Computers (PCs) & Mobile Devices:**
- နေ့စဉ်သုံးနေတဲ့ Laptop, Desktop, Tablet နဲ့ Smart Phone တွေပါ။ တစ်ယောက်တည်း သုံးဖို့ ရည်ရွယ်ပါတယ် ။
2. **Servers (ဆာဗာများ):**
- ဒါက အင်တာနက်ရဲ့ အဓိကပါ။ Facebook ပေါ်တင်လိုက်တဲ့ ဓာတ်ပုံတွေ၊ YouTube ဗီဒီယိုတွေကို သိမ်းထားပေးတဲ့ ၂၄ နာရီ မပိတ်တမ်းဖွင့်ထားတဲ့ ကွန်ပျူတာကြီးတွေ ဖြစ်ပါတယ်။ (အရင်ခေတ်က Mainframe လို့ ခေါ်တဲ့ အရာတွေရဲ့ နေရာမှာ အစားထိုးလာတာပါ) ။

3. **Supercomputers (စူပါ ကွန်ပျူတာများ):**
- ကမ္ဘာပေါ်မှာ အမြန်ဆုံး ကွန်ပျူတာတွေပါ။ မိုးလေဝသ ခန့်မှန်းတာ၊ ငလျင်အန္တရာယ် တွက်ချက်တာ၊ ဆေးပညာ စမ်းသပ်တာတွေမှာ သုံးပါတယ် ။

1.4 ကွန်ပျူတာ ဘယ်လို အလုပ်လုပ်သလဲ? (Summary)

Chapter 1 ရဲ့ အနှစ်ချုပ်အနေနဲ့ - သူငယ်ချင်းဆီ
ကို Messenger ကနေ "Hello" လို့ ပို့လိုက်တဲ့ ဖြစ်စဉ်ကို ကြည့်ရအောင်။

1. **Input:** ဒီဘက်ဖုန်း Screen ပေါ်က 'H-e-l-l-o' ဆိုတဲ့ စာလုံးတွေကို နှိပ်လိုက်တယ် (Keyboard/Touchscreen က အလုပ်လုပ်တယ်) ။
2. **Process:** ဖုန်းရဲ့ CPU က ဒီစာလုံးတွေကို အင်တာနက်ကနေ ပို့ရမယ့် ဒေတာအဖြစ် ပြောင်းလဲတွက်ချက်လိုက်တယ် ။
3. **Output:** သူငယ်ချင်းရဲ့ ဖုန်း Screen ပေါ်မှာ "Hello" ဆိုပြီး ပေါ်လာတယ် ။

Chapter 2: Data Representation

(အချက်အလက်များကို ကွန်ပျူတာ နားလည်အောင် ပုံဖော်ခြင်း)

ကွန်ပျူတာတစ်လုံးဟာ အပြင်ပန်းကြည့်ရင် ရှုပ်ထွေးတယ်လို့ ထင်ရပေမယ့် အတွင်းပိုင်း အလုပ်လုပ်ပုံက အလွန်ရိုးရှင်းပါတယ်။ ကွန်ပျူတာတွေ၊ ဖုန်းတွေဟာ လူတွေလို စာလုံးတွေ၊ ဓာတ်ပုံတွေ၊ သီချင်းတွေကို တိုက်ရိုက် နားမလည်ပါဘူး။ သူတို့ နားလည်တာက **"လျှပ်စစ် မီးပွင့် (On)"** နဲ့ **"လျှပ်စစ် မီးပိတ် (Off)"** ဆိုတဲ့ အခြေအနေ နှစ်ခုတည်းသာ ဖြစ်ပါတယ် ။

ဒီ On/Off သဘောတရားကို သချိၤနည်းအရ **1 နှင့် 0** လို့ သတ်မှတ်ပြီး **Binary System (ဒွိကိန်းစနစ်)** လို့ ခေါ်ပါတယ် ။

2.1 Bit နှင့် Byte ဆိုတာ ဘာလဲ? (Units of Data)

ကွန်ပျူတာ နည်းပညာမှာ မကြာခဏ ကြားရတဲ့ Bit နဲ့ Byte ဆို
တာ အချက်အလက် ပမာဏကို တိုင်းတာတဲ့ ယူနစ် (Units) တွေ ဖြစ်ပါတယ်။

- **Bit (b):** Binary Digit ရဲ့ အတိုကောက်ပါ။ 0 သို့မဟုတ် 1 တစ်လုံးတည်း ကို ဆိုလိုပါတယ်။ ဒါဟာ ကွန်ပျူတာ အချက်အလက်တွေရဲ့ အသေး ဆုံး ယူနစ် ဖြစ်ပါတယ်။
- **Byte (B):** Bit အရေအတွက် ၈ လုံး စုစည်းထားတာကို 1 Byte လို့ခေါ်ပါ တယ် (ဥပမာ - 01000001 ဆိုရင် 1 Byte ပါ)။ စာလုံး
  တစ်လုံး (ဥပမာ 'A') ကို သိမ်းဆည်းဖို့ ပုံမှန်အားဖြင့် 1 Byte နေရာယူပါ တယ် ။

ခေတ်သစ် မှတ်ဉာဏ်ပမာဏများ:

ဖုန်းဝယ်တဲ့အခါ Storage 128 GB, 256 GB ဆိုပြီး ပြောကြတာ ကြားဖူးမှာပါ။
ဒီပမာဏတွေက ဘယ်လောက်များသလဲဆိုတာ ကြည့်ရအောင် -

- **Kilobyte (KB):** ခန့်မှန်းခြေ စာလုံးရေ ၁,၀၀၀ (တိကျစွာပြောရရင် 1,024 Bytes)။
- **Megabyte (MB):** ခန့်မှန်းခြေ ၁ သန်း (သီချင်းတစ်ပုဒ်ဟာ ပျမ်းမျှ 5 MB လောက် ရှိပါတယ်)။
- **Gigabyte (GB):** ခန့်မှန်းခြေ ၁ ဘီလီယံ (ရုပ်ရှင်ကားကောင်းတစ်ကား ဟာ 2 GB ကနေ 4 GB လောက် ရှိနိုင်ပါတယ်)။
- **Terabyte (TB):** ခန့်မှန်းခြေ ၁ ထရီလီယံ (လက်ရှိ
  ခေတ် ကွန်ပျူတာ Hard Disk တွေရဲ့ ပမာဏပါ)။

2.2 Binary System (0 နှင့် 1 ရေတွက်ပုံ)

လူသားတွေက **Decimal (base-10)** လို့ခေါ်တဲ့ ဒသမစနစ်ကို သုံးပါတယ်။ (**0** ကနေ **9** အထိ ဂဏန်း ၁၀ လုံး ရှိပါတယ်)။ ကွန်ပျူတာကတော့ **Binary (base-2)** စနစ်ကို သုံးပါတယ်။ (**0** နဲ့ **1** ဂဏန်း ၂ လုံးပဲ ရှိပါတယ်) ။ ရေတွက်ပုံ ကွာခြားချက်ကို ဇယားဖြင့် လေ့လာကြည့်ပါ -

Decimal (လူတွေရေတွက်ပုံ)	Binary (ကွန်ပျူတာသိမ်းဆည်းပုံ)	ရှင်းပြချက်
0	0	လျှပ်စစ်ပိတ် (Off)
1	1	လျှပ်စစ်ပွင့် (On)
2	10	နေရာတစ်နေရာ တိုးသွားသည် (Decimal တွင် 9 ပြီးရင် 10 ဖြစ်သကဲ့သို့)
3	11	
4	100	နောက်ထပ် နေရာတစ်ခု ထပ်တိုးသည်
5	101	

2.3 Hexadecimal (ဂဏန်းအကြီးများကို အတိုကောက် မှတ်သားခြင်း)

ကွန်ပျူတာ ပရိဂရမ်မာတွေအတွက် Binary ဂဏန်းတွေ (ဥပမာ - 101101101010) ကို ဖတ်ရတာ အရမ်းရှုပ်ထွေးပြီး မှတ်မိဖို့ ခက်ခဲပါတယ် ။ ဒါကြောင့် **Hexadecimal (base-16)** စနစ်ကို အတိုကောက်အနေနဲ့ သုံးကြပါတယ်။

- **0** ကနေ **9** အထိကို ဂဏန်းအတိုင်း သုံးပါတယ်။
- **10** ကနေ **15** အထိကိုတော့ **A, B, C, D, E, F** ဆိုပြီး စာလုံးတွေနဲ့ ကိုယ်စားပြုပါတယ်။

လက်တွေ့ ဥပမာ (အရောင်များ):
Website တွေ၊ ဒီဇိုင်းဆော့ဖ်ဝဲတွေမှာ အရောင်တွေကို Hex Code နဲ့ သတ်မှတ်ပါတယ်။

- အနီရောင် (Red) = #FF0000
- အစိမ်းရောင် (Green) = #00FF00
- အဖြူရောင် (White) = #FFFFFF (F ဆိုတာ 15 ဖြစ်ပြီး တန်ဖိုးအမြင့်ဆုံး/အလင်းဆုံး ဖြစ်လို့ အဖြူရောင် ဖြစ်သွားတာပါ)။

2.4 စာလုံးများ၊ ဓာတ်ပုံများနှင့် အသံများ

ကွန်ပျူတာက ဂဏန်းတွေကိုပဲ နားလည်တယ်ဆိုရင် ကျွန်တော်တို့ ရိုက်ပို့နေတဲ့ **"Hello"** ဆိုတဲ့ စာတွေ၊ **Selfie** ပုံတွေကို သူဘယ်လိုသိမ်းဆည်းသလဲ?
(a) စာလုံးများ (Text Encoding)

ကီးဘုတ်ပေါ်က စာလုံးတစ်လုံးချင်းစီမှာ သတ်မှတ်ထားတဲ့ နံပါတ်ကုဒ်တွေ ရှိပါတယ်။

- **ASCII Code:** အရင်ခေတ်က အင်္ဂလိပ်စာလုံးတွေအတွက်ပဲ သုံးခဲ့တဲ့ စနစ်ပါ။ (ဥပမာ - 'A' ဆိုရင် နံပါတ် 65)။
- **Unicode:** ယနေ့ခေတ်မှာတော့ မြန်မာစာအပါအဝင် ကမ္ဘာ့ဘာသာစကားအားလုံးနဲ့ Emoji (😀) တွေပါ ရအောင် Unicode စနစ်ကို သုံးပါတယ်။ 'A' ကို 65 (Binary: 01000001) အဖြစ် ပြောင်းလဲသိမ်းဆည်းပါတယ်။

(b) ဓာတ်ပုံများ (Digital Images)

ဖုန်းထဲက ဓာတ်ပုံတစ်ပုံကို အကြီးချဲ့ကြည့်လိုက်ရင် လေးထောင့်အကွက်လေးတွေ တွေ့ရမှာပါ။ အဲ့ဒါကို **Pixel** လို့ခေါ်ပါတယ်။ Pixel တစ်ကွက်ချင်းစီမှာ သူ့ရဲ့ အရောင်ကို ကိုယ်စားပြုတဲ့ Binary ကုဒ်တစ်ခုစီ ရှိပါတယ်။ အဲ့ဒီ ကုဒ်တွေအားလုံးကို စုစည်းလိုက်တဲ့အခါ ဓာတ်ပုံတစ်ပုံ ဖြစ်လာပါတယ်။

(c) အသံများ (Digital Audio)

အသံလှိုင်းတွေကို တစ်စက္ကန့်မှာ အကြိမ်ပေါင်းများစွာ (Sampling) တိုင်းတာပြီး ဂဏန်းတွေအဖြစ် ပြောင်းလဲမှတ်သားပါတယ်။ ဥပမာ - သီချင်းဖွင့်လိုက်တာနဲ့ ကွန်ပျူတာက သိမ်းထားတဲ့ သန်းပေါင်းများစွာသော 0 နဲ့ 1 တွေကို အသံလှိုင်းအဖြစ် ပြန်ပြောင်းပေးလိုက်တာ ဖြစ်ပါတယ်။

Chapter 3: Information and Logic

(ကွန်ပျူတာ၏ ဆုံးဖြတ်ချက်ချမှတ်ပုံ)

ကွန်ပျူတာတစ်လုံးဟာ အလွန်ရှုပ်ထွေးတဲ့ အလုပ်တွေကို လုပ်နေတယ်လို့ ထင်ရပေမယ့် တကယ်တမ်းမှာတော့ သူတို့ဟာ ရိုးရှင်းတဲ့ မေးခွန်းတွေကို **"ဟုတ်တယ် (True)"** သို့မဟုတ် **"မဟုတ်ဘူး (False)"** လို့ ဖြေရင်းနဲ့ပဲ အလုပ်လုပ်ကြတာပါ။ ဒီသဘောတရားကို **Proposition Logic** လို့ခေါ်ပါတယ်။

3.1 Proposition Logic (အမှန်/အမှား ဆုံးဖြတ်ခြင်း)

Proposition (အဆိုပြုချက်) ဆိုတာ "အမှန်" သို့မဟုတ် "အမှား" တန်ဖိုးတစ်ခုခု ရှိနေတဲ့ စာကြောင်းကို ဆိုလိုပါတယ်။ ဥပမာ -

- "အခု မိုးရွာနေတယ်" (ဒါက အမှန် ဖြစ်နိုင်သလို၊ အမှားလည်း ဖြစ်နိုင်ပါတယ်)။
- "x = 5" (ဒါက Proposition တစ်ခုပါ)။
- “မင်္ဂလာပါ" လို့ပြောလိုက်တယ်ထား (ဒါက အမှန်/အမှား ခွဲခြားလို့မရတဲ့ အတွက် Proposition မဟုတ်ပါဘူး)။

ကွန်ပျူတာလောကမှာတော့ ဒီ အမှန် (True) နဲ့ အမှား (False) ကို **1 နဲ့ 0** အနေနဲ့ မှတ်သားပါတယ်။

3.2 Logical Operators (ဆုံးဖြတ်ချက်ချသည့် သင်္ကေတများ)

အခြေအနေတစ်ခုထက်မက ရှုပ်ထွေးလာတဲ့အခါ အဆိုပြုချက်တွေကို ချိတ်ဆက်ပြီး ဆုံးဖြတ်ရပါတယ်။ အဲ့ဒီလိုချိတ်ဆက်ဖို့ အဓိက **Logical Operators** (၃) ခုကို သုံးပါတယ်။

(a) AND Gate (နှင့်)

နှစ်ခုလုံး "အမှန်" ဖြစ်မှ ရလဒ်က "အမှန်" ဖြစ်မယ့် Logic ပါ။

- **သဘောတရား:** "A လည်းမှန်၊ B လည်းမှန်မှ အလုပ်လုပ်မယ်။"
- **သင်္ကေတ:** `&&` သို့မဟုတ် `.` (အစက်)

လက်တွေ့ဥပမာ (Login ဝင်ခြင်း):

- Facebook အကောင့်ထဲဝင်ဖို့ (၁) ဖုန်းနံပါတ် မှန်ရမယ် **AND** (၂) Password မှန်ရမယ်။
- တစ်ခုခုမှားတာနဲ့ ဝင်လို့မရပါဘူး (False ဖြစ်သွားပါမယ်)။

ဖုန်းနံပါတ်	Password	ဝင်ခွင့် (Result)
မှန်	မှား	မရ (False)
မှား	မှန်	မရ (False)
မှန်	မှန်	ရသည် (True)

(b) OR Gate (သို့မဟုတ်)

တစ်ခုခု "အမှန်" ဖြစ်တာနဲ့ ရလဒ်က "အမှန်" ဖြစ်မယ့် Logic ပါ။

- **သဘောတရား:** "A ဖြစ်ဖြစ်၊ B ဖြစ်ဖြစ် တစ်ခုခုမှန်ရင် အလုပ်လုပ်မယ်။"
- **သင်္ကေတ:** `||` သို့မဟုတ် `+` (အပေါင်း)

လက်တွေ့ ဥပမာ (ငွေပေးချေခြင်း):

- Supermarket မှာ ပိုက်ဆံရှင်းမယ်ဆိုရင် (၁) Cash (ငွေသား) နဲ့ပေးလို့ရတယ် **OR** (၂) KPay နဲ့ပေးလို့လည်းရတယ်။
- တစ်ခုခုပါလာရင် ဈေးဝယ်လို့ရပါပြီ။

Cash ပါလာလား	Mobile Pay ရှိလား	ဝယ်လို့ရလား (Result)
မပါ	မပါ	မရ (False)
ပါ	မပါ	ရသည် (True)
မပါ	ပါ	ရသည် (True)
ပါ	ပါ	ရသည် (True)

(c) NOT Gate (ပြောင်းပြန်)

ရှိနေတဲ့ အခြေအနေကို ပြောင်းပြန်လှန်လိုက်တဲ့ Logic ပါ။

- **သဘောတရား:** "အမှန်ဆိုရင် အမှားဖြစ်စေ၊ အမှားဆိုရင် အမှန်ဖြစ်စေ။"
- **သင်္ကေတ:** `!` သို့မဟုတ် `¬`

လက်တွေ့ ဥပမာ (Mute လုပ်ခြင်း):

- အသံဖွင့်ထားတာကို NOT လုပ်လိုက်ရင် "အသံပိတ် (Mute)" ဖြစ်သွားမယ်။
- Dark Mode ဖွင့်ထားတာကို NOT လုပ်လိုက်ရင် "Light Mode" ဖြစ်သွားမယ်။

Dark Mode	Light Mode
on	off
off	on

(d) XOR Gate (Exclusive OR – တစ်ခုတည်းသာ)

ဒါက OR နဲ့ ဆင်တူပေမဲ့ အနည်းငယ်ကွဲပြားပါတယ်။ "နှစ်ခုလုံးမှန်ရင် လက်မခံဘူး၊ တစ်ခုတည်းမှန်မှ လက်ခံမယ်" ဆိုတဲ့ Logic ပါ။

သဘောတရား: "A နဲ့ B မတူမှ အမှန်ပြမယ်။ တူနေရင် အမှားပြမယ်။"

လက်တွေ့ ဥပမာ (မီးခလုတ်):

လှေကားအတက်အဆင်းမှာ တပ်ထားတဲ့ မီးခလုတ် (2-way switch) လိုမျိုးပါ။ အောက်ကခလုတ်ဖွင့်ထားပြီး၊ အပေါ်ကခလုတ် ပိတ်ထားမှ မီးလင်းမယ်။ နှစ်ခုလုံးဖွင့်ထားရင် (သို့) နှစ်ခုလုံးပိတ်ထားရင် မီးမလင်းပါဘူး။

မီးခလုတ် ၁	မီးခလုတ် ၂	မီးသီး

ပိတ်	ပိတ်	မီးမလင်း
ဖွင့်	ပိတ်	မီးလင်း
ပိတ်	ဖွင့်	မီးလင်း
ဖွင့်	ဖွင့်	မီးမလင်း

(e) NAND Gate & NOR Gate (ဆန့်ကျင်ဘက်ဂိတ်များ)

ဒါကတော့ မူရင်း AND နဲ့ OR ကို NOT (ပြောင်းပြန်) လုပ်လိုက်တာနဲ့ အတူတူပါပဲ။

- **NAND (Not AND):** AND နဲ့ပြောင်းပြန်ပါ။ နှစ်ခုလုံးမှန်မှ **False** ဖြစ်ပြီး၊ ကျန်အချိန်တွေမှာ **True** ပါ။
- **NOR (Not OR):** OR နဲ့ပြောင်းပြန်ပါ။ နှစ်ခုလုံးမှားမှ **True** ဖြစ်ပြီး၊ တစ်ခုခုမှန်တာနဲ့ **False** ဖြစ်ပါတယ်။

3.3 Combinations (ပေါင်းစပ်အသုံးပြုခြင်း)

ဒီ Logic အခြေခံတွေကို ပေါင်းစပ်ပြီး ပိုရှုပ်ထွေးတဲ့ ဆုံးဖြတ်ချက်တွေကို ချမှတ်နိုင်ပါတယ်။

ဥပမာ – YouTube Video ကြည့်ခြင်း Video ကြည့်ခွင့်ရဖို့ ကွန်ပျူတာက ဒီလို စဉ်းစားပါတယ် -

1. (အင်တာနက်ဖွင့်ထားလား **AND** ဖုန်းဘေရိုလား?) -> ရှိမှ ဆက်သွားမယ်။
2. (Video က Free လား **OR** ပိုက်ဆံပေးပြီး ဝယ်ထားလား?) -> တစ်ခုခုဟုတ်ရင် ကြည့်လို့ရပြီ။

ဒီလို ဆုံးဖြတ်ချက်အဆင့်ဆင့်ကို **Logic Circuits** တွေ၊ **Algorithms** တွေ နဲ့ တည်ဆောက်ထားတာ ဖြစ်ပါတယ်။

Chapter 4: Hardware

(ကွန်ပျူတာ၏ ရုပ်ပိုင်းဆိုင်ရာ တည်ဆောက်ပုံ)

Hardware ဆိုတာ "လူခန္ဓာကိုယ်" နဲ့ တူတယ်လို့ ပြောခဲ့ပါတယ်။ ဒီအခန်းမှာ တော့ ခန္ဓာကိုယ်ရဲ့ အရေးကြီးဆုံး အစိတ်အပိုင်းတွေဖြစ်တဲ့ ဦးနှောက် (CPU)၊ မှတ်ဉာဏ် (Memory) နဲ့ အာရုံခံအစိတ်အပိုင်းများ (Input/Output) အကြောင်း ကို အသေးစိတ် ကြည့်ကြပါမယ် ။

4.1 Processor (ကွန်ပျူတာ၏ ဦးနှောက်)

Processor သို့မဟုတ် CPU (Central Processing Unit) ဆိုတာ ကွန်ပျူတာ စနစ်တစ်ခုလုံးရဲ့ ခေါင်းဆောင် ဖြစ်ပါတယ် ။ သူ့မှာ အဓိက အစိတ်အပိုင်း ကြီး (၂) ခု ပါဝင်ပါတယ် -

1. **Control Unit (CU):**

- ဒါက လူရဲ့ ဗဟိုအာရုံကြောစနစ် (Central Nervous System) နဲ့ တူပါတယ် ။
- သူက တွက်ချက်ခြင်း မလုပ်ပါဘူး။ ဘယ်သူ့ကို ဘာလုပ်ရမယ်ဆိုတာ လိုက်လံညွှန်ကြားပါတယ် ။ မှတ်ဉာဏ်ထဲက ညွှန်ကြားချက်တွေကို ယူတယ်၊ ဘာလုပ်ရမလဲ ဘာသာပြန်တယ်၊ ပြီး ရင် သက်ဆိုင်ရာ နေရာတွေကို ခိုင်းစေပါတယ် ။

2. Arithmetic and Logic Unit (ALU):

- ဒါကတော့ တကယ့် "တွေးခေါ်တဲ့ အပိုင်း" (Thinking Section) ဖြစ်ပါတယ် ။
- ဂဏန်းပေါင်းနှုတ်တာတွေ (Arithmetic)၊ ဟုတ်/မဟုတ် ဆုံးဖြတ်တာတွေ (Logic) အားလုံးကို ဒီနေရာမှာ လုပ်ပါတယ် ။

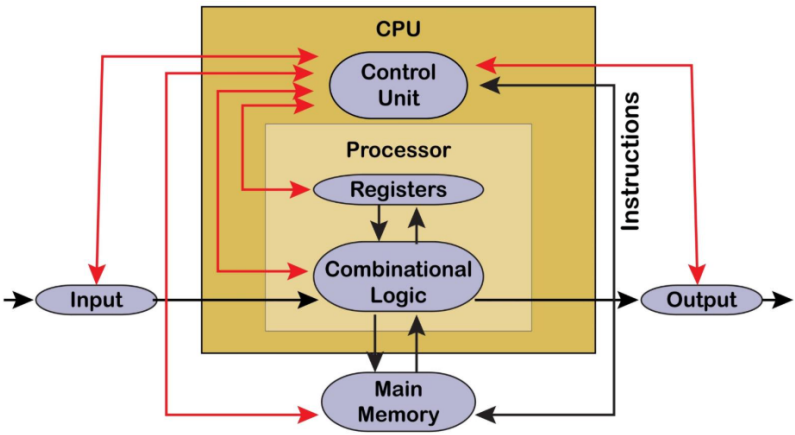


image - Shutterstock

CPU အလုပ်လုပ်ပုံ (Machine Cycle): CPU က အလုပ်တစ်ခုလုပ်တိုင်း ဒီအဆင့် (၃) ဆင့်အတိုင်း လုပ်ပါတယ် -

- **Fetch:** လုပ်ရမယ့် အလုပ်ကို Memory ထဲက သွားယူတယ်။
- **Decode:** ဘာလုပ်ခိုင်းတာလဲဆိုတာ နားလည်အောင် ဘာသာပြန်တယ်။
- **Execute:** လက်တွေ့ လုပ်ဆောင်တယ်။

4.2 Memory (မှတ်ဉာဏ်)

ကွန်ပျူတာမှာ မှတ်ဉာဏ်သိမ်းဆည်းတဲ့ နေရာက တစ်မျိုးတည်း မဟုတ်ပါဘူး။ အသုံးပြုပုံပေါ်မူတည်ပြီး အဆင့်ဆင့် ခွဲခြားထားပါတယ် ။

Memory Hierarchy (အဆင့်ဆင့် တည်ဆောက်ပုံ):

1. **Registers:** CPU အတွင်းထဲမှာတင် ရှိတဲ့ အလွန်သေးငယ်ပြီး အမြန်ဆုံး မှတ်ဉာဏ်ပါ။ တွက်ချက်နေတုန်း ဂဏန်းတွေ(1 နဲ့ 0 တွေ)ကို ခဏကိုင်ထားတဲ့ နေရာပါ ။
2. **Main Memory (RAM):**

- "အလုပ်လုပ်နေတဲ့ စားပွဲခုံ" နဲ့ တူပါတယ်။ ဖွင့်ထားတဲ့ App တွေ၊ Run နေတဲ့ System တွေ ဒီပေါ်မှာ ရှိနေပါတယ် ။
- သူ့ရဲ့ ထူးခြားချက်ကတော့ စက်ပိတ်လိုက်တာနဲ့ အထဲက အချက်အလက်တွေ အကုန်ပျောက်သွားခြင်း (Volatile) ဖြစ်ပါတယ် ။

3. **Auxiliary Storage (Secondary Storage):**

- စက်ပိတ်လိုက်လည်း မပျောက်တဲ့ "ရေရှည် မှတ်ဉာဏ်" တွေ ဖြစ်ပါတယ် ။

4.3 Auxiliary Storage Devices (အချက်အလက် သိမ်းဆည်းသည့် ကိရိယာများ)

1. **Hard Disk Drive (HDD):** အရင်ခေတ်က သံလိုက်ပြားဝိုင်းတွေနဲ့ လည်ပတ်ပြီး Data သိမ်းတဲ့ ပုံစံပါ။ ဈေးသက်သာပြီး ပမာဏ အများကြီး (TB နဲ့ချီပြီး) သိမ်းနိုင်ပေမယ့် နှေးကွေးပါတယ်။
2. **Solid State Drive (SSD):** ယနေ့ခေတ် ကွန်ပျူတာအသစ်တွေမှာ သုံးပါတယ်။ သူက HDD လို စက်ဝိုင်းပြားတွေ မလည်တော့ဘဲ Chip တွေနဲ့ သိမ်းတာကြောင့် အသံမမြည်ဘူး၊ အများကြီး ပိုမြန်ပါတယ်။

3. **Cloud Storage:** Google Drive, iCloud စတာတွေက ကိုယ့်စက်ထဲမှာ မဟုတ်ဘဲ အင်တာနက်ပေါ်က ဆာဗာတွေမှာ လှမ်းသိမ်းထားတာ ဖြစ်ပါတယ်။

4.4 Input / Output Devices

Input Devices (ကွန်ပျူတာထဲသို့ အချက်အလက် ထည့်သွင်းသည့် ကိရိယာများ)

လူတွေ နားလည်တဲ့ အချက်အလက် (စာ၊ အသံ၊ ပုံရိပ်) တွေကို ကွန်ပျူတာ နားလည်တဲ့ 0 နဲ့ 1 အဖြစ် ပြောင်းပေးသူတွေ ဖြစ်ပါတယ် ။

- **Touchscreen:** မျက်နှာပြင်ကို ထိတာနဲ့ Input ဝင်သွားပါတယ်။
- **Sensors:** ဖုန်းမှာပါတဲ့ Face ID (မျက်နှာဖတ်), Gyroscope (ဖုန်းစောင်းတာသိ), GPS (နေရာသိ) စတာတွေပါ။
- Keyboard, Mouse, Scanner။

Output Devices (ရလဒ်များကို) ပြသ/ဖော်ပြသည့် ကိရိယာများ)

ကွန်ပျူတာက တွက်ချက်လိုက်တဲ့ 0 နဲ့ 1 ရလဒ်တွေကို လူတွေ နားလည်အောင် ပြန်ပြပေးသူတွေ ဖြစ်ပါတယ် ။

- **Display:** အရင်က TV အိုးကြီးတွေနဲ့တူတဲ့ CRT မဟုတ်တော့ဘဲ၊ ပါးလွှာပြီး ရုပ်ထွက်ကောင်းတဲ့ **LED, OLED Screen** တွေ ဖြစ်လာပါပြီ ။
- **Printers:** စာရွက်ပေါ် ပုံနှိပ်တာအပြင်၊ ပလပ်စတစ်ရည်တွေကို ပုံဖော်ပေးတဲ့ **3D Printer** တွေပါ ပေါ်လာပါပြီ ။

Chapter 5: Data Structures & Algorithms

(အချက်အလက် ဖွဲ့စည်းပုံနှင့် လုပ်ဆောင်ချက်များ)

5.1 Data Structure ဆိုတာ ဘာလဲ?

Data Structure ဆိုတာ အချက်အလက် (Data) တွေကို ကွန်ပျူတာထဲမှာ စနစ်တကျ သိမ်းဆည်းတဲ့ ပုံစံ (Model) တစ်ခုဖြစ်ပါတယ် ။ ဥပမာ - ဝန်ထမ်းတစ်ယောက်ရဲ့ နာမည်၊ အသက်၊ လစာ စတာတွေကို စုစည်းပြီး မှတ်တမ်း (Record) တစ်ခုအနေနဲ့ သိမ်းဆည်းတာမျိုး ဖြစ်ပါတယ် ။ အသုံးအများဆုံး Data Structure ပုံစံတွေကို နေ့စဉ်မြင်ကွင်းတွေနဲ့ နှိုင်းယှဉ်လေ့လာကြည့်ရအောင်။

5.2 Linear Data Structures (တန်းစီနေသော ပုံစံများ)

Data တွေကို တစ်ခုပြီးတစ်ခု တန်းစီသိမ်းဆည်းတဲ့ ပုံစံတွေဖြစ်ပါတယ်။

(a) Array (အကွက်လိုက် သိမ်းဆည်းခြင်း)

ဒါက အရိုးရှင်းဆုံးပုံစံပါ။ Data တွေကို နံပါတ်စဉ်တပ်ထားတဲ့ အကွက်လေးတွေထဲမှာ တစ်ခုချင်းစီ ထည့်သိမ်းတာပါ ။

- **ဥပမာ:** ကြက်ဥကတ် တစ်ကတ်မှာ ကြက်ဥ ၁၀ လုံး ထည့်စရာပါရင်၊ အကွက် အမှတ် ၁၊ အမှတ် ၂ ဆိုပြီး နေရာသတ်မှတ်ထားသလိုပါပဲ။
- **အားသာချက်:** နံပါတ် (Index) သိတာနဲ့ တန်းပြီး ယူလို့ရတယ်။ ဥပမာ - Student[1] ဆိုရင် နံပါတ် ၁ ကျောင်းသားကို တန်းခေါ်လို့ရတယ် ။

(b) Stack (ထပ်ထားခြင်း – LIFO)

ဒါက ပန်းကန်ဆေးဖို့ ဘေစင်မှာ ပန်းကန်တွေ ဆင့်ထပ်ထားတာနဲ့ တူပါတယ် ။

- **စည်းကမ်း: "Last-In, First-Out (LIFO)"** (နောက်ဆုံးမှ ရောက်လာတဲ့လူ၊ အရင်ဆုံး ပြန်ထွက်ရမယ်) ။
- **လက်တွေ့ ဥပမာ:** * ပန်းကန်ဆေးတဲ့အခါ အပေါ်ဆုံး (နောက်ဆုံးမှ တင်ထားတဲ့) ပန်းကန်ကို အရင်ဆေးရတယ်။
  - ကွန်ပျူတာမှာ စာရိုက်ရင်းမှားလို့ **Undo (Ctrl+Z)** နှိပ်လိုက်ရင် နောက်ဆုံးလုပ်ခဲ့တဲ့ အလုပ်ကို အရင်ဖျက်ပေးတာ Stack သဘောတရားပါပဲ။

(c) Queue (တန်းစီခြင်း – FIFO)

ဒါကတော့ လက်မှတ်ဝယ်ဖို့ လူတွေ တန်းစီနေတာနဲ့ တူပါတယ် ။

- **စည်းကမ်း: "First-In, First-Out (FIFO)"** (အရင်ရောက်တဲ့လူ၊ အရင်သွားရမယ်) ။
- **လက်တွေ့ ဥပမာ:** * Printer တစ်လုံးတည်းကို လူအများကြီးက လှမ်း Print ရင် အရင်ခိုင်းတဲ့သူရဲ့ စာရွက် အရင်ထွက်လာပါတယ်။
  - Spotify မှာ သီချင်းတွေကို "Add to Queue" လုပ်ထားရင် အရင်ထည့်ထားတဲ့သီချင်း အရင်လာပါတယ်။

5.3 Non-Linear Data Structures (အဆင့်ဆင့် ပုံစံများ)

Data တွေက အမြဲတမ်း တန်းစီနေတာ မဟုတ်ပါဘူး။ တစ်ခါတစ်လေ အဆင့်ဆင့် ခွဲထွက်သွားတာမျိုးလည်း ရှိပါတယ်။

(a) Tree (သစ်ပင်ပုံစံ)

ဒါက မိသားစု ဆွေစဉ်မျိုးဆက်ပြတဲ့ ဇယား (Family Tree) သို့မဟုတ် ကုမ္ပဏီတစ်ခုရဲ့ ရာထူးကြီးစဉ်ငယ်လိုက် ဖွဲ့စည်းပုံ (Organization Chart) နဲ့ တူပါတယ်

။ အပေါ်မှာ အကြီးဆုံး (Root) တစ်ခုရှိမယ်၊ အောက်မှာ အကိုင်းအခက်တွေ (Branches) ခွဲထွက်သွားမယ် ။

- **လက်တွေ့ ဥပမာ:** ကွန်ပျူတာထဲက Folder စနစ် ဖြစ်ပါတယ်။ My Computer ထဲမှာ Local Disk (C:)၊ အဲ့ဒီအထဲမှာ Users၊ Desktop၊ Photos ဆိုပြီး အဆင့်ဆင့် ခွဲသွားတာ Tree ပုံစံပါပဲ။

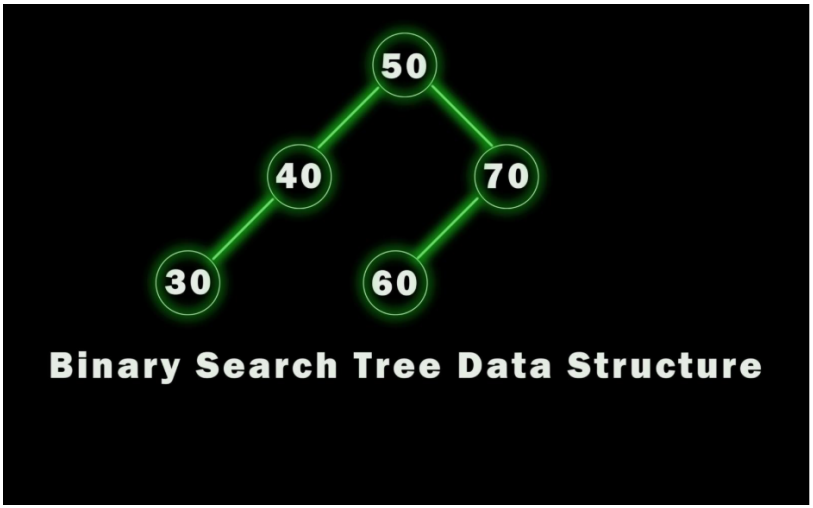


image - Shutterstock

(b) Graph (ကွန်ရက်ပုံစံ)

ဒါကတော့ အဆင့်ဆင့် မဟုတ်ဘဲ တစ်ခုနဲ့တစ်ခု ရှုပ်ထွေးစွာ ချိတ်ဆက်နေတဲ့ ပုံစံပါ ။

- **လက်တွေ့ ဥပမာ:** * **Social Network:** Facebook မှာ မင်းနဲ့ မင်းသူငယ်ချင်း၊ သူငယ်ချင်းရဲ့ သူငယ်ချင်း စသဖြင့် ချိတ်ဆက်နေပုံ။

- **Maps:** လေယာဉ်ခရီးစဉ်တွေမှာ မြို့တစ်မြို့နဲ့ တစ်မြို့ လမ်းကြောင်းတွေ ချိတ်ဆက်ထားပုံ ။

5.4 Algorithms (ပြဿနာဖြေရှင်းနည်းများ)

Data တွေကို သိမ်းပြီးရင် ပြန်ရှာဖို့ လိုပါတယ်။ ပြဿနာတစ်ခုကို ဖြေရှင်းဖို့ အဆင့်ဆင့် လုပ်ဆောင်ရမယ့် လုပ်ငန်းစဉ်ကို **Algorithm** လို့ခေါ်ပါတယ် ။ ရှာဖွေခြင်း (Searching) အတွက် လူသုံးအများဆုံး Algorithm နှစ်ခုကို ယှဉ်ကြည့်ရအောင်။

1. Linear Search (တစ်ခုချင်း လိုက်ရှာခြင်း)

- **နည်းလမ်း:** စာအုပ်ပုံထဲမှာ စာအုပ်တစ်အုပ်ရှာသလိုမျိုး အပေါ်ဆုံးက နေ အောက်ဆုံးထိ တစ်အုပ်ချင်းစီ ဖယ်ပြီး ရှာတာပါ။
- **အားနည်းချက်:** စာအုပ် ၁၀၀၀ ရှိရင် ၁၀၀၀ လုံး ရှာရနိုင်တဲ့အတွက် ကြာပါတယ် ။

2. Binary Search (ထက်ဝက်ချိုး ရှာခြင်း)

- **နည်းလမ်း:** ဖုန်းစာအုပ် (သို့) အဘိဓာန် (Dictionary) မှာ စာလုံးရှာသလိုမျိုးပါ။
  - စာအုပ်ကို အလယ်ကနေ လှပ်လိုက်မယ်။
  - ကိုယ်ရှာချင်တာက ညာဘက်အခြမ်းမှာ ရှိမယ်ဆိုရင် ဘယ်ဘက်အခြမ်းကို မရှာတော့ဘူး။
  - ကျန်တဲ့ ညာဘက်အခြမ်းကို ထပ်ပြီး တစ်ဝက်ချိုးမယ် ။
  - အဲ့လိုထပ်ထပ်ပြီး မတွေ့မချင်း။
- **အားသာချက်:** အလွန်မြန်ပါတယ်။ လူနာမည် ၂၅,၀၀၀ ထဲက တစ်ယောက်ကို ရှာတာတောင် ၁၅ ကြိမ်လောက်ပဲ ရှာရပါတယ် ။

- **မှတ်ချက်:** Binary Search သုံးချင်ရင် Data တွေက အစဉ်လိုက် (Sorted) ဖြစ်နေမှ ရပါမယ် (ဥပမာ - A to Z စီထားမှ ရမယ်) ။

Chapter 6: Memory Organization

(မှတ်ဉာဏ် ဖွဲ့စည်းပုံ)

6.1 မိတ်ဆက် (Introduction)

ကွန်ပျူတာ CPU တစ်လုံး အလုပ်လုပ်ဖို့အတွက် ညွှန်ကြားချက် (Instructions) နဲ့ အချက်အလက် (Data) တွေကို သိမ်းဆည်းပေးမယ့် နေရာ လိုအပ်ပါတယ်။ အဲ့ဒါကို Memory လို့ခေါ်ပါတယ် ။
Memory ကို လေ့လာတဲ့အခါ ရှုထောင့် (၂) မျိုးနဲ့ ကြည့်နိုင်ပါတယ် -

1. **Logical Organization:** Memory ကို Software ရှုထောင့်က နေ ဘယ်လိုမြင်သလဲ (ဥပမာ - လိပ်စာနံပါတ် တပ်ထားတဲ့ အကွက်လေးများအဖြစ် မြင်ခြင်း)။
2. **Physical Organization:** ဟာ့ဒ်ဝဲ ရှုထောင့်ကနေ ဘယ်လို တည်ဆောက်ထားသလဲ (ဥပမာ - Chip ပြားတွေ၊ လျှပ်စစ်ပတ်လမ်းတွေ)။

6.2 Logical Organization (ဆော့ဖ်ဝဲ မြင်ကွင်း)

ကွန်ပျူတာ စနစ်ထဲမှာ Memory ကို အခန်းပေါင်းများစွာပါတဲ့ "တိုက်တန်းရှည်ကြီး" တစ်ခုလို မြင်ကြည့်နိုင်ပါတယ်။

- **Cells & Locations:** Memory မှာ အကွက်လေးတွေ (Cells) သန်းချီ ပြီး ပါဝင်ပါတယ်။ အကွက်တစ်ကွက်ချင်းစီမှာ Data တွေကို သိမ်းဆည်း ပါတယ် ။
- **Addressing (လိပ်စာတပ်ခြင်း):** အိမ်တိုင်းမှာ အိမ်နံပါတ်ရှိသလို၊ Memory အကွက်တိုင်းမှာလည်း သူ့ သက်ဆိုင်ရာ လိပ်စာ (Unique Address) ရှိပါတယ် ။ CPU က Data လိုချင်ရင် "Address 100 မှာရှိတဲ့ Data ကို ပေးပါ" ဆို ပြီး လိပ်စာနဲ့ လှမ်းခေါ်ရပါတယ် ။

Memory Read/Write Operation (ဖတ်ခြင်း/ရေးခြင်း လုပ်ငန်းစဉ်)

CPU က Memory နဲ့ အလုပ်လုပ်တဲ့အခါ အထူးမှတ်ဉာဏ် (Registers) နှစ်ခု ကို အသုံးပြုပါတယ်

1. **MAR (Memory Address Register):** ကိုယ်သွားချင်တဲ့ "လိပ်စာ" ကို ထည့်တဲ့နေရာ။
2. **MDR (Memory Data Register):** သယ်ချင်တဲ့ "Data" ကို ထည့်တဲ့ နေရာ။

ဥပမာ – Memory ထဲက Data ကို ဖတ်ချင်ရင် (Read Operation):

1. **Address:** လိုချင်တဲ့ လိပ်စာကို MAR ထဲ ထည့်လိုက်တယ်။
2. **Signal:** "Read" (ဖတ်မယ်) ဆိုပြီး အမိန့်ပေးလိုက်တယ်။
3. **Copy:** အဲ့ဒီလိပ်စာမှာရှိတဲ့ Data က MDR ထဲကို ရောက်လာတယ်။

ဥပမာ – Memory ထဲကို Data ထည့်ချင်ရင် (Write Operation):

1. **Address:** ထည့်ချင်တဲ့ လိပ်စာကို MAR ထဲ ထည့်တယ်။
2. **Data:** ထည့်ချင်တဲ့ Data ကို MDR ထဲ ထည့်တယ်။

3. **Signal:** "Write" (ရေးမယ်) ဆိုပြီး အမိန့်ပေးလိုက်ရင် Memory ထဲ ရောက်သွားပါပြီ။

6.3 Physical Organization (ဟာ့ဒ်ဝဲ တည်ဆောက်ပုံ)

ခေတ်ဟောင်း ကွန်ပျူတာတွေမှာ သံလိုက်နည်းပညာတွေ (Magnetic Drum, Ferrite Core) ကို သုံးခဲ့ကြပေမယ့် ၊ ယနေ့ခေတ်မှာတော့ **Semiconductor (Silicon Chips)** တွေကို အသုံးပြုလာကြပါပြီ ။
ခေတ်သစ် Memory အမျိုးအစားများကို အောက်ပါအတိုင်း ခွဲခြားနိုင်ပါတယ် -

(a) RAM (Random Access Memory)

ဒါက ကွန်ပျူတာဖွင့်ထားတုန်း အလုပ်လုပ်တဲ့ ယာယီမှတ်ဉာဏ် ဖြစ်ပါတယ်။ သူ့မှာ နှစ်မျိုးကွဲပါတယ် -

1. **Dynamic RAM (DRAM):**

- ဒါက ကျွန်တော်တို့ ကွန်ပျူတာတွေမှာ သုံးနေတဲ့ RAM Stick တွေ (ဥပမာ - DDR4, DDR5) ဖြစ်ပါတယ်။
- သူ့ရဲ့ သဘောသဘာဝက လျှပ်စစ်အားကို အချိန်နဲ့အမျှ ပြန်ပြန်ဖြည့်ပေးရပါတယ် (Refresh လုပ်ရတယ်)။ မဟုတ်ရင် Data ပျောက်သွားတတ်ပါတယ်။
- စျေးသက်သာပြီး ပမာဏအများကြီး (8GB, 16GB) ထုတ်လုပ်နိုင်ပါတယ် ။

2. **Static RAM (SRAM):**

- ဒါက အရမ်းမြန်တဲ့ Memory ပါ။ Refresh လုပ်စရာမလိုပါဘူး။
- ဒါပေမဲ့ စျေးအရမ်းကြီးပြီး နေရာယူတဲ့အတွက် သူ့ကို **CPU Cache** (L1, L2 Cache) အနေနဲ့ပဲ အနည်းငယ် ထည့်သုံးကြပါတယ် ။

(b) ROM (Read Only Memory)

- စက်ရုံကတည်းက အသေရေးထည့်ပေးလိုက်တဲ့ Data တွေပါ ။
- ဥပမာ - ကွန်ပျူတာ စဖွင့်ဖွင့်ချင်း တက်လာတဲ့ စာလုံးတွေ၊ BIOS/ UEFI System တွေဟာ ROM ထဲမှာ ရှိပါတယ်။
- အရင်ခေတ်က ဖျက်လို့မရပေမယ့်၊ အခုခေတ်မှာတော့ **Flash Memory** (EEPROM) နည်းပညာကြောင့် ပြန်ပြင်ရေးလို့ ရလာပါပြီ (Firmware Update လုပ်တယ်ဆိုတာ ဒါကိုပြောတာပါ) ။

6.4 Speed and Capacity (အမြန်နှုန်းနှင့် ပမာဏ)

Memory တစ်ခုရဲ့ စွမ်းဆောင်ရည်ကို တိုင်းတာတဲ့အခါ အဓိက (၂) ချက် ရှိပါတယ် -

1. **Capacity (ပမာဏ):**

- Data ဘယ်လောက်များများ ဆံ့သလဲ။ (KB, MB, GB, TB) ။
- ယနေ့ခေတ် စံနှုန်း: RAM ဆိုရင် 8GB မှ 32GB အထိ၊ SSD ဆိုရင် 256GB မှ 1TB အထိ။

2. **Access Time (ကြာချိန်):**

- CPU က လှမ်းတောင်းလိုက်တဲ့အချိန်နဲ့ Data ရောက်လာတဲ့အချိန် ကြား ဘယ်လောက်ကြာလဲ ။
- **RAM:** Nanoseconds(ns) ပိုင်းပဲ ကြာပါတယ် (1 ns = တစ် စက္ကန့်၏ တစ်ဘီလီယံပုံတစ်ပုံ)။
- **Hard Disk:** Milliseconds (ms) ပိုင်း ကြာတတ်ပါတယ် ။ (ဒါကြောင့် RAM က Hard Disk ထက် အဆပေါင်းသိန်းချီ ပိုမြန်တာပါ)။

6.5 Error Detection (အမှားရှာဖွေခြင်း)

Memory ထဲကို Data ကူးပြောင်းတဲ့အခါ တစ်ခါတစ်လေ လျှပ်စစ်အနှောင့်အယှက်ကြောင့် 0 နဲ့ 1 ပြောင်းပြန် ဖြစ်သွားတတ်ပါတယ်။ ဒါကို စစ်ဆေးဖို့ **Parity Bit** ဆိုတဲ့ နည်းပညာကို သုံးပါတယ် ။

- Data 8-bit ကိုသိမ်းတဲ့အခါ နောက်ဆုံးမှာ အပို 1-bit (Parity bit) ထပ်ထည့်လိုက်ပါတယ်။
- Data ထဲမှာ 1 အရေအတွက်က "စုံ" လား "မ" လား ဆိုတာကို ကြည့်ပြီး Parity bit မှာ မှတ်ထားလိုက်ပါတယ်။
- ပြန်ဖတ်တဲ့အချိန်မှာ ကိုက်ညီမှုမရှိရင် Error ရှိနေမှန်း သိနိုင်ပါတယ် ။

Chapter 7: System Software & Operating Systems

(စနစ်ပိုင်းဆိုင်ရာ ဆော့ဖ်ဝဲများနှင့် လည်ပတ်ရေးစနစ်)

7.1 System Software ဆိုတာ ဘာလဲ?

ကွန်ပျူတာ ဆော့ဖ်ဝဲတွေကို အဓိက (၂) မျိုး ခွဲခြားနိုင်ပါတယ်:

1. **Application Software:** အသုံးပြုသူ လုပ်ချင်တဲ့ အလုပ်တစ်ခုအတွက် သုံးတာပါ။ (ဥပမာ - စာစီဖို့ Word၊ သီချင်းနားထောင်ဖို့ Spotify၊ ဂိမ်းဆော့ဖို့ Mobile Legends)။

2. **System Software:** ကွန်ပျူတာ ဟာ့ဒ်ဝဲတွေကို ထိန်းချုပ်မောင်းနှင်ပေးပြီး Application တွေ အလုပ်လုပ်နိုင်အောင် ကူညီပေးတဲ့ အခြေခံ ဆော့ဖ်ဝဲတွေ ဖြစ်ပါတယ် ။

System Software မှာ **Operating System (OS)**၊ **Language Processors** (ဘာသာပြန်သူများ) နဲ့ **Utility Programs** (အထွေထွေသုံးများ) ဆိုပြီး အဓိက ပါဝင်ပါတယ် ။

7.2 Operating System (OS) ၏ အခန်းကဏ္ဍ

Operating System (OS) ဆိုတာ ကွန်ပျူတာတစ်လုံးရဲ့ "မန်နေဂျာ" ဖြစ်ပါတယ် ။ CPU, RAM, Storage စတဲ့ အရင်းအမြစ် (Resources) တွေကို ဘယ်သူ့ကို ဘယ်လောက်ပေးသုံးမလဲဆိုတာ သူက စီမံခန့်ခွဲပါတယ် ။
OS ရဲ့ အဓိက အလုပ် (၄) ခုကို လေ့လာကြည့်ရအောင်:

(a) Processor Management (ဦးနှောက်ကို စီမံခြင်း)

CPU က တစ်ချိန်တည်းမှာ အလုပ်တစ်ခုပဲ လုပ်နိုင်ပါတယ် ။ ဒါပေမဲ့ ကျွန်တော်တို့က သီချင်းနားထောင်ရင်း စာရိုက်လို့ရနေတာ ဘာလို့လဲ?

- **Multiprogramming & Time Sharing:** OS က အလုပ်တွေအားလုံးကို အချိန်ပိုင်းလေးတွေ (Time Slices) ခွဲပြီး CPU ကို အလှည့်ကျ ပေးသုံးနေလို့ ဖြစ်ပါတယ် ။ ဒီဖြစ်စဉ်က အရမ်းမြန်တဲ့အတွက် လူတွေက "ပြိုင်တူလုပ်နေတယ်" လို့ ထင်ရခြင်း ဖြစ်ပါတယ်။

(b) Memory Management (မှတ်ဉာဏ်ကို စီမံခြင်း)

RAM ဆိုတဲ့ စားပွဲခုံပေါ်မှာ ဘယ် App ကို နေရာဘယ်လောက်ပေးမလဲဆိုတာ OS က ဆုံးဖြတ်ပါတယ် ။

- **Virtual Memory (အတုယောင် မှတ်ဉာဏ်):** RAM မလောက်တော့တဲ့အခါ (ဥပမာ - Game ကြီးကြီးဆော့နေချိန်) OS က Hard

Disk ထဲက နေရာလွတ်ကို RAM အဖြစ် ခေတ္တယူသုံးပါတယ် ။ ဒါကြောင့် RAM ပြည့်သွားရင် စက်က လေးသွားတတ်တာ ဖြစ်ပါတယ်။

(c) Device Management (ကိရိယာများကို စီမံခြင်း)

မတူညီတဲ့ Printer တွေ၊ WiFi ကိရိယာတွေ၊ Mouse တွေကို ကွန်ပျူတာက ဘယ်လိုသိသလဲ? **Device Drivers** လို့ခေါ်တဲ့ ကြားခံပရိုဂရမ်လေးတွေကတဆင့် OS က ဆက်သွယ်မောင်းနှင်ပါတယ် ။

(c) File Management (ဖိုင်များကို စီမံခြင်း)

Hard Disk ပေါ်မှာ 0 နဲ့ 1 တွေအဖြစ် ပြန့်ကျဲနေတဲ့ Data တွေကို ကျွန်တော်တို့ မြင်ရတဲ့ **Folder** တွေ၊ **File** တွေအဖြစ် သစ်ပင်ပုံစံ (Tree Structure) စနစ်တကျ စီပေးတာ OS ဖြစ်ပါတယ် ။

7.3 OS အမျိုးအစားများနှင့် ခေတ်သစ် ဥပမာများ

ကွန်ပျူတာလောကမှာ အသုံးများတဲ့ OS တွေကို ဒီလို ခွဲခြားနိုင်ပါတယ် :

1. **Personal Computer OS (တစ်ကိုယ်ရည်သုံး):**

- **Microsoft Windows:** ရုံးသုံး၊ ကျောင်းသုံး အသုံးအများသုံး OS ဖြစ်ပါတယ်။
- **macOS:** Apple ကွန်ပျူတာများတွင် သုံးပြီး ဒီဇိုင်းပိုင်း ကောင်းမွန်ပါတယ်။
- **Linux:** ပရိုဂရမ်မာတွေနဲ့ Server တွေမှာ အသုံးများပါတယ်။ Open Source (အခမဲ့) ဖြစ်ပါတယ်။

2. **Mobile OS (ဖုန်းသုံး):**

- **Android:** Google က ထုတ်လုပ်ပြီး ဖုန်းအမျိုးအစားစုံမှာ သုံးပါတယ်။

- **iOS:** Apple iPhone များအတွက် သီးသန့် ဖြစ်ပါတယ်။

3. **Network & Server OS:**

- အင်တာနက်ပေါ်က Web Server တွေ၊ Database တွေ run ဖို့ သီးသန့် ထုတ်လုပ်ထားတဲ့ OS တွေ (ဥပမာ - Windows Server, Ubuntu Server) ဖြစ်ပါတယ် ။

7.4 Client-Server နှင့် Cloud Computing

ခေတ်သစ်ကွန်ပျူတာ စနစ်တွေမှာ စက်တစ်လုံးတည်း အလုပ်မလုပ်တော့ပါဘူး။ ကွန်ရက်ချိတ်ဆက်ပြီး အလုပ်လုပ်ကြပါတယ်။

Client–Server Model:

- **Client (အသုံးပြုသူ):** ဝန်ဆောင်မှုကို လှမ်းတောင်းတဲ့သူ (ဥပမာ - ဖုန်းကနေ YouTube ကြည့်တာ၊ facebook posts list ကို refresh လုပ်လိုက်တာ၊ တစ်ခုခု login ဝင်လိုက်တာ)။
- **Server (ဝန်ဆောင်မှုပေးသူ):** တောင်းတာကို ပေးပို့တဲ့သူ (ဥပမာ - YouTube ရဲ့ ဗီဒီယိုတွေ သိမ်းထားတဲ့ ကွန်ပျူတာကြီး၊ facebook posts တွေနဲ့ account တွေကိုသိမ်းထားတာ)။

Cloud Computing:

- အရင်ခေတ်က **Time Sharing** (ကွန်ပျူတာကြီးတစ်လုံးကို အများမျှသုံးခြင်း) ကနေ ပြောင်းလဲလာတာပါ။
- အခုခေတ်မှာတော့ Google Drive, iCloud ဆိုတာတွေက "Cloud" လို့ ခေါ်ပေမယ့် တကယ်တော့ ကမ္ဘာတစ်နေရာက Server ကြီးတွေထဲမှာ ကိုယ့် Data တွေကို လှမ်းသိမ်းထားခြင်းသာ ဖြစ်ပါတယ်။

Chapter 8: Programming Languages

(ပရိုဂရမ်ရေးသားခြင်း ဘာသာစကားများ)

8.1 Programming Language အဆင့်ဆင့်

ကွန်ပျူတာ ဘာသာစကားတွေကို လူတွေနားလည်နိုင်မှု အပေါ်မူတည်ပြီး အဆင့် (၃) ဆင့် ခွဲခြားနိုင်ပါတယ် ။

(a) Machine Language (ကွန်ပျူတာသာနားလည်တဲ့ ဘာသာစကား)

ဒါက အနိမ့်ဆုံးအဆင့် (Lowest Level) ဖြစ်ပြီး ကွန်ပျူတာ ဟာ့ဒ်ဝဲက တိုက်ရိုက် နားလည်တဲ့ ဘာသာစကားဖြစ်ပါတယ် ။

- **ပုံစံ:** 0 နဲ့ 1 တွေချည်းပဲ ပါဝင်ပါတယ် (Binary String) ။
- **အခက်အခဲ:** လူသားတွေအတွက် ဖတ်ရှုဖို့၊ ပြင်ဆင်ဖို့ အလွန်ခက်ခဲပါတယ် ။ စက်တစ်လုံးနဲ့ တစ်လုံး (CPU Architecture မတူရင်) ကုဒ်တွေ မတူကြပါဘူး ။

(b) Assembly Language (လူနားလည်တဲ့ ခက်ခဲတဲ့ဘာသာစကား)

Machine Language ထက် နည်းနည်းပိုမြင့်ပြီး 0 နဲ့ 1 အစား အတိုကောက် စာလုံး (Mnemonics) တွေကို သုံးပါတယ် ။

- **ပုံစံ:** ADD (ပေါင်း), LOAD (ဆွဲတင်), STORE (သိမ်း) စသဖြင့် ရေးပါတယ် ။
- **အသုံးပြုမှု:** ဟာ့ဒ်ဝဲကို တိုက်ရိုက်ထိန်းချုပ်ဖို့ လိုအပ်တဲ့ အပိုင်းတွေမှာ သုံးပါတယ် (ဥပမာ - Device Driver ရေးခြင်း)။ ဒါပေမဲ့ ဒီကုဒ်

ကို စက်နားလည်အောင် **Assembler** ဆိုတဲ့ ပရိဂရမ်နဲ့ ပြန်ပြောင်းပေးရပါတယ် ။

(c) High–Level Language (အဆင့်မြင့် ဘာသာစကား)

ဒါက လူတွေပြောတဲ့ အင်္ဂလိပ်စာကြောင်းတွေနဲ့ အနီးစပ်ဆုံးတူအောင် ဖန်တီးထားတဲ့ ဘာသာစကားတွေ ဖြစ်ပါတယ် ။

- **ပုံစံ:** if x > 10: print("Hello") ဆိုပြီး ဖတ်လိုက်တာနဲ့ နားလည်နိုင်ပါတယ်။
- **အားသာချက်:** ရေးရလွယ်တယ်၊ ပြင်ရလွယ်တယ် ။ စက်အမျိုးအစားမရွေး အလုပ်လုပ်နိုင်ပါတယ် (Machine Independent) ။
- **ဥပမာ:** Python, Java, C++, JavaScript။

8.2 Language Processors (programming language တစ်ခုနဲ့တစ်ခုပြောင်းပေးတဲ့ program)

High-Level Language နဲ့ ရေးထားတဲ့ ကုဒ် (Source Code) တွေကို စက်နားလည်တဲ့ Machine Code ဖြစ်အောင် ဘာသာပြန်ပေးဖို့ လိုပါတယ် ။ ဘာသာပြန်ပုံပေါ်မူတည်ပြီး (၂) မျိုး ကွဲပြားပါတယ် ။

(a) Preprocessor (ကြိုတင် ပြင်ဆင်ပေးသူ)

Compiler ဆီ မရောက်ခင် Source Code ကို ကြိုတင် ပြင်ဆင်မွမ်းမံပေးတဲ့ ပရိဂရမ် ဖြစ်ပါတယ်။

- သူက High Level Language တစ်ခုကနေ အခြား High Level Language တစ်ခုကို ပြောင်းပေးတာမျိုး (သို့မဟုတ်) ကုဒ်တွေကို ချဲ့ထွင် (Expand) ပေးတာမျိုး လုပ်ဆောင်ပါတယ် ။
- ဥပမာ - ဘာသာစကားအသစ်တစ်ခုကို ဒီဇိုင်းဆွဲတဲ့အခါ ဒါကိုသုံးလေ့ရှိပါတယ် ။

(b) Compiler (machine language ထိထုတ်ပေးတဲ့ program)

Source Code တစ်ခုလုံးကို အစကနေ အဆုံးထိ တစ်ခါတည်း ဘာသာပြန်ပြီး .exe (Executable File) အဖြစ် ထုတ်ပေးပါတယ် ။

- **အားသာချက်:** Run တဲ့အခါ အရမ်းမြန်ပါတယ် (ပြန်ပြီး ဘာသာပြန်နေစရာ မလိုတော့လို့ပါ)။
- **အားနည်းချက်:** ကုဒ်ပြင်လိုက်တိုင်း အစကနေ ပြန်ပြီး Compile လုပ်ပေးရပါတယ်။
- **ဥပမာ:** C, C++။

(c) Interpreter (တစ်ကြောင်းချင်း ဘာသာပြန်)

Source Code ကို တစ်ကြောင်းချင်း ဖတ်တယ်၊ ချက်ချင်း ဘာသာပြန်တယ်၊ ချက်ချင်း အလုပ်လုပ်တယ် ။

- **အားသာချက်:** Error တက်ရင် ဘယ်နားမှာလဲ ချက်ချင်းသိရတယ်၊ ပြင်ရလွယ်တယ် ။
- **အားနည်းချက်:** Compiler လောက် မမြန်ပါဘူး (Run တိုင်း ဘာသာပြန်နေရလို့ပါ)။
- **ဥပမာ:** Python, JavaScript, PHP။

(c) Assembler (Assembly to Machine)

Assembly Language (Low-level language) နဲ့ ရေးထားတဲ့ ကုဒ်တွေကို စက်နားလည်တဲ့ Machine Code (0 နဲ့ 1) အဖြစ် ပြောင်းပေးပါတယ် ။

8.3 Programming Standards (စံနစ်တကျ ရေးသားခြင်း)

ပရိုဂရမ်တစ်ခုဟာ "အလုပ်လုပ်ရင် ပြီးတာပဲ" လို့ ယူဆလို့မရပါဘူး။ နောက်လူတွေ ဝင်ဖတ်ရင် နားလည်အောင်၊ ပြင်ဆင်ရလွယ်အောင် စံနစ်တကျ ရေးသားဖို့ လိုပါတယ် ။

(a) Structured & Modular Programming

ပရိုဂရမ်အကြီးကြီးတွေကို ဒီတိုင်းရေးမယ့်အစား အပိုင်းငယ် (Modules) လေးတွေ ခွဲပြီး ရေးသားသင့်ပါတယ် ။ ဒါမှ အပိုင်းတစ်ပိုင်း ပြင်ချင်ရင် ကျန်တာတွေကို ထိခိုက်မှု မရှိမှာ ဖြစ်ပါတယ် ။

(b) Comments & Indentation

- **Comments:** ကုဒ်ထဲမှာ ဒီအကြောင်းက ဘာလုပ်တာလဲဆိုတာ မှတ်ချက် ရေးခဲ့သင့်ပါတယ်။ (Python မှာ # နဲ့ ရေးပါတယ်)။
- **Indentation:** စာကြောင်းတွေကို ညီညီညာညာ ခွာရေးတာက ကုဒ်ကို ဖတ်ရလွယ်ကူစေပါတယ် ။

Chapter 9: Network Technology

(ကွန်ရက် နည်းပညာနှင့် အင်တာနက်)

9.1 Network ဆိုတာ ဘာလဲ?

ကွန်ပျူတာတစ်လုံးတည်း သီးသန့်မနေဘဲ တစ်လုံးနဲ့တစ်လုံး ချိတ်ဆက်ပြီး အချက်အလက်တွေ ဖလှယ်နိုင်အောင် လုပ်ထားတာကို **Network** (ကွန်ရက်) လို့ခေါ်ပါတယ်။

အရွယ်အစားပေါ် မူတည်ပြီး အဓိက (၂) မျိုး ခွဲခြားနိုင်ပါတယ်:

- **LAN (Local Area Network):** အိမ်တစ်အိမ်၊ ရုံးခန်းတစ်ခန်း အတွင်းမှာပဲ ချိတ်ဆက်ထားတဲ့ ကွန်ရက်အသေးစားပါ။ (ဥပမာ - အိမ်က WiFi မှာ ဖုန်းတွေ၊ TV တွေ ချိတ်ထားတာ LAN ပါပဲ)။
- **WAN (Wide Area Network):** မြို့တွေ၊ နိုင်ငံတွေ ဖြတ်ကျော်ပြီး ချိတ်ဆက်ထားတဲ့ ကွန်ရက်အကြီးစားပါ။ (ဥပမာ - အင်တာနက်)။

9.2 Network Topologies (ကွန်ရက် ချိတ်ဆက်ပုံစံများ)

ကွန်ပျူတာတွေကို ကြိုးတွေနဲ့ ချိတ်ဆက်တဲ့အခါ ပုံစံအမျိုးမျိုး ရှိပါတယ် ။

(a) Star Topology (ကြယ်ပွင့်ပုံစံ)

ဗဟိုချက်မတစ်ခု (Central Hub) ရှိပြီး ကျန်တဲ့စက်တွေအားလုံးက အဲ့ဒီအလယ်ကို လာချိတ်တဲ့ ပုံစံပါ။

- **လက်တွေ့ ဥပမာ:** အိမ်က **WiFi Router** က အလယ်မှာ ရှိတယ်၊ ဖုန်းတွေ၊ Laptop တွေအားလုံးက အဲ့ဒီ Router ကို ဝိုင်းချိတ်ကြတယ်။
- **အားသာချက်:** ဖုန်းတစ်လုံး ပျက်သွားရင် ကျန်တဲ့လူတွေ အင်တာနက် သုံးလို့ရနေတုန်းပါပဲ။

(b) Mesh Topology (ပင့်ကူအိမ်ပုံစံ)

စက်တိုင်းက တခြားစက်တိုင်းနဲ့ တိုက်ရိုက်ချိတ်ဆက်ထားတဲ့ ပုံစံပါ။

- **လက်တွေ့ ဥပမာ: Starlink** (ဂြိုဟ်တု အင်တာနက်) သို့မဟုတ် ရုံးကြီးတွေမှာသုံးတဲ့ **Mesh WiFi** စနစ်တွေပါ။
- **အားသာချက်:** လမ်းကြောင်းတစ်ခု ပြတ်တောက်သွားရင် တခြားလမ်းကြောင်းကနေ အလိုအလျောက် ပြောင်းသွားလို့ ယုံကြည်စိတ်ချရမှု အမြင့်ဆုံးပါ ။

9.3 Transmission Media (ချိတ်ဆက်သည့် နည်းလမ်းများ)

ကွန်ပျူတာတွေ ဘယ်လို စကားပြောကြသလဲ? ကြိုးနဲ့ ချိတ်မလား၊ လေထဲက သွားမလား?

(a) Wired (ကြိုးဖြင့် ချိတ်ဆက်ခြင်း)

- **Fiber Optic Cable:** အလင်းတန်းကို သုံးပြီး Data ပို့တာဖြစ်လို့ အလွန်မြန်ပါတယ်။ ပင်လယ်ရေအောက် ကေဘယ်ကြိုးတွေ (Submarine Cables) ဟာ ဒီနည်းပညာကို သုံးပြီး နိုင်ငံတွေကို ချိတ်ဆက်ထားတာပါ ။
- **Twisted Pair (LAN ကြိုး):** ရုံးတွေမှာ ကွန်ပျူတာအနောက်ကို ထိုးထည့်တဲ့ ခေါင်းပြာပြာနဲ့ ကြိုးတွေ ဖြစ်ပါတယ် ။

(b) Wireless (ကြိုးမဲ့ ချိတ်ဆက်ခြင်း)

- **Wi-Fi:** ရေဒီယိုလှိုင်း (Radio Waves) ကို သုံးပြီး နီးနီးနားနား ချိတ်ဆက်တာပါ ။

- **Cellular Data (4G/5G):** ဖုန်းတာဝါတိုင်တွေကနေတဆင့် အဝေးကြီးကို လှမ်းချိတ်တာပါ။

9.4 The Internet & Connection (အင်တာနက် အလုပ်လုပ်ပုံ)

အင်တာနက်ဆိုတာ ကမ္ဘာပေါ်မှာရှိတဲ့ ကွန်ပျူတာ သန်းပေါင်းများစွာကို ချိတ်ဆက်ထားတဲ့ "ကွန်ရက်ကြီးများ၏ ကွန်ရက်" (Network of Networks) ဖြစ်ပါတယ် ။

(a) IP Address (လိပ်စာ)

အင်တာနက်ပေါ်ရောက်ရင် စက်တိုင်းမှာ ကိုယ်ပိုင်လိပ်စာတစ်ခု ရှိရပါတယ်။ အဲ့ဒါကို **IP Address** လို့ခေါ်ပါတယ်။

- ဥပမာ - 202.52.68.46 ။
- စာတိုက်က စာပို့ရင် အိမ်နံပါတ်လိုပါပဲ။ Data တွေက ဒီနံပါတ်အတိုင်း အရောက်သွားပါတယ်။

(b) DNS (အမည်ပြောင်း စနစ်)

ဂဏန်းတွေ အများကြီးကို လူတွေ မမှတ်မိနိုင်ပါဘူး။ ဒါကြောင့် google.com ဆိုတဲ့ နာမည်ကို သုံးကြပါတယ်။

- **DNS (Domain Name System)** ဆိုတာ "ဖုန်းစာအုပ်" (Phonebook) နဲ့ တူပါတယ်။
- မင်းက google.com လို့ ရိုက်လိုက်ရင်၊ DNS Server က 142.250.1.1 ဆိုတဲ့ IP Address ကို ရှာပြီး ပြောင်းပေးလိုက်ပါတယ် ။

(c) Client–Server & Cloud

- **Client-Server Model:** မင်းရဲ့ဖုန်း (Client) က Facebook ကို ဖွင့်လိုက်တဲ့အခါ Facebook ရဲ့ Server (ကွန်ပျူတာကြီး) ဆီကနေ Data တွေကို လှမ်းတောင်းပါတယ် ။
- **Cloud Computing:** "Cloud" ဆိုတာ တိမ်တိုက်ပေါ်မှာ ရှိတာမဟုတ်ပါဘူး။ Google တို့၊ Microsoft တို့ရဲ့ ကုမ္ပဏီကြီးတွေထဲက Server ကြီးတွေကို ခေါ်တာပါ။ မင်းရဲ့ ဓာတ်ပုံတွေကို ကိုယ့်ဖုန်းထဲ (Hard Disk) မှာ မသိမ်းဘဲ၊ သူတို့ရဲ့ Server Disk တွေထဲမှာ လှမ်းသိမ်းထားတာ ဖြစ်ပါတယ်။

*"ဤစာအုပ်များသည် တက္ကသိုလ်များတွင် သင်ကြားပို့ချနေသော Computer Science သင်ရိုးညွှန်းတမ်း (Curriculum) များကို အခြေခံထားပါသည်။ သို့သော် ယနေ့ခေတ် နည်းပညာများနှင့် ကိုက်ညီစေရန်အတွက် Python ဘာသာစကား၊ ခေတ်မီ Hardware နည်းပညာများနှင့် လက်တွေ့လုပ်ငန်းခွင်သုံး ဥပမာများကို ဖြည့်စွက် ပြုပြင်ရေးသားထားခြင်း ဖြစ်ပါသည်။

မူရင်းသင်ရိုးကို ပြုစုခဲ့ကြသော ဆရာ/ဆရာမ များအားလုံးကို လေးစားစွာဖြင့် Credit ပေးပါသည်။"*

Yan Naing Lin
Software Engineer