컴퓨터 개념 및 실습

기말 고사 review

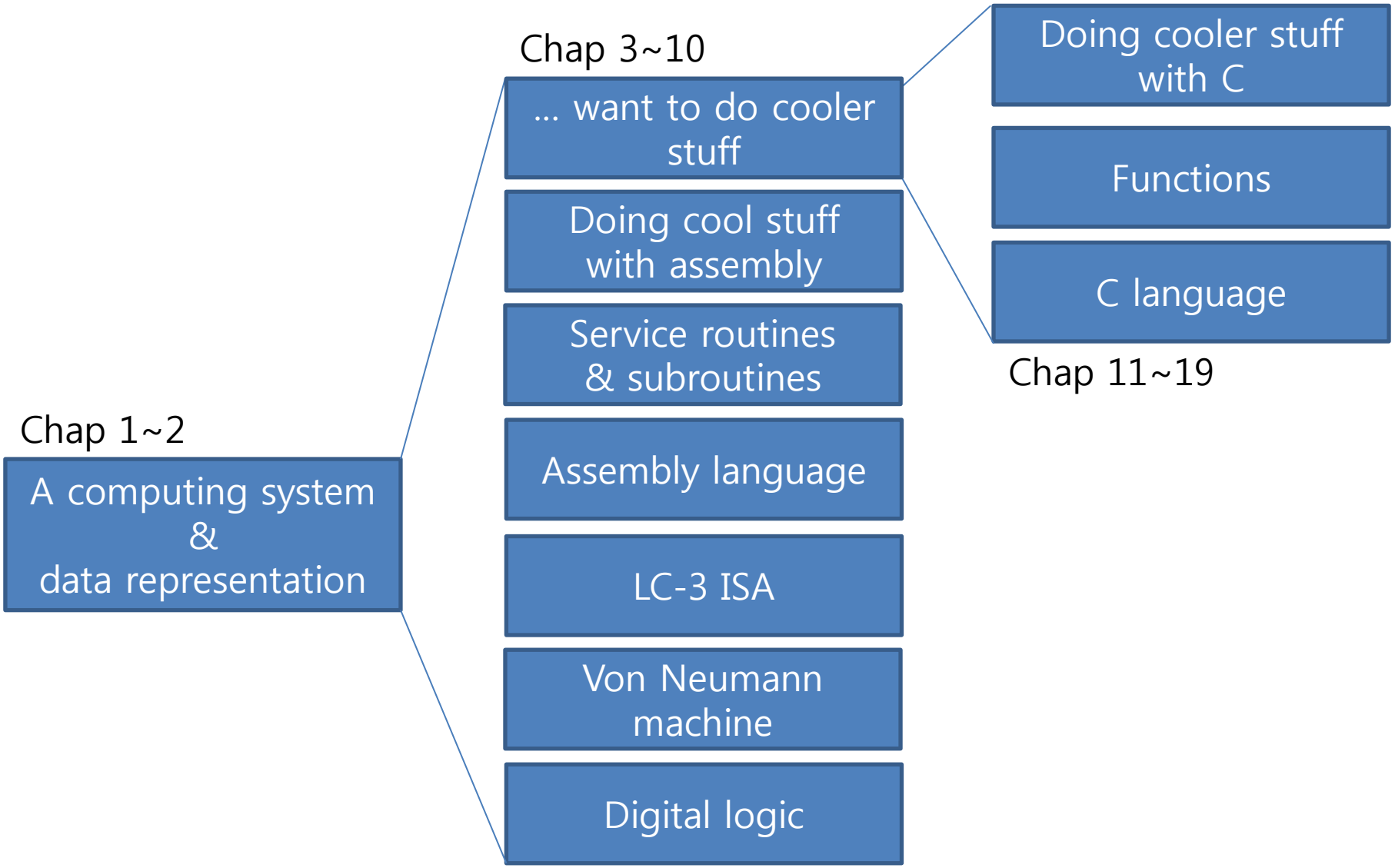
Sorting results

Sort	Size	Compare count
Insert $O(n^2)$	5	4 (lucky data set)
	9	24
Select $O(n^2)$	5	10
	9	36
Bubble $O(n^2)$	5	15
	9	40
Merge $O(n \cdot \log n)$	14	39
Quick $O(n \cdot \log n)$	15	45 (unlucky pivot)

The entire course in a nutshell

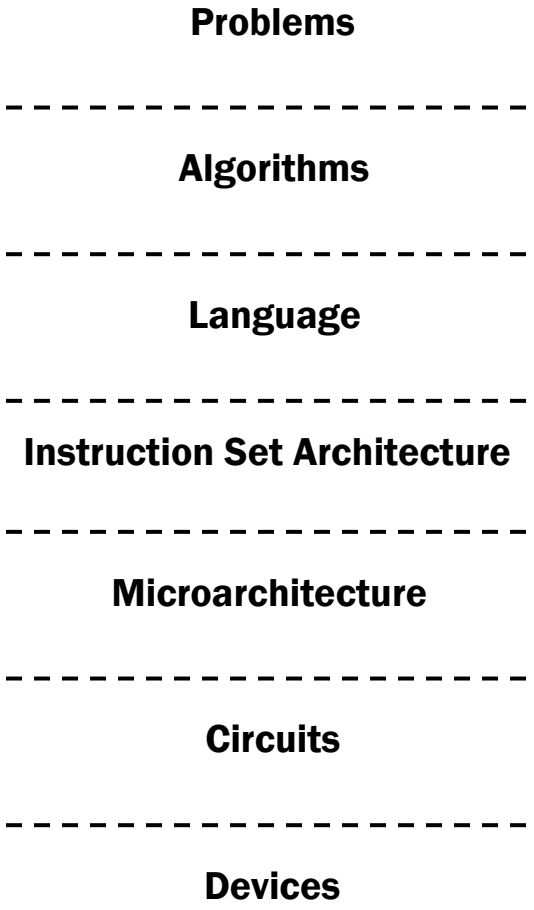
- The underlying structure of a computer
  - Chap 1~Chap 10
- Programming in a high-level language
  - Chap 11~Chap 19

My interpretation of the textbook



Important ideas from Chap. 1

- Abstraction
- HW vs. SW
- Universal computing device



* Fig. 1.6 in textbook

Important ideas from Chap. 2

- Integer representation
  - Sign-magnitude
  - 1's complement
  - 2's complement
- Operations
  - Arithmetic: add, subtract, sign-ext, overflow
  - Logical: and, or, not… nand, nor, xor, xnor
- Other data representations
  - Floating point
  - ASCII

Important ideas from Chap. 3

- Transistor
- Logic gates
  - Truth table
  - DeMorgan's Law, and other laws
- Combinational logic
  - Decoder
  - Mux
  - Adder
- Storage elements
  - Latch
  - Gated latch
  - Register
  - Memory
- Sequential logic
  - Flip-flop
  - Finite state machine

Important ideas from Chap. 4

- Von Neumann model
  - Memory
  - Processing unit
  - Control unit
  - Input
  - Output
- Inst. processing
  - Fetch inst.
  - Decode
  - Eval. addr.
  - Fetch operands
  - Execute
  - Store

Important ideas from Chap. 5

- LC-3 ISA
  - Memory, registers
  - Instructions
- LC-3 instructions
  - Operate
  - Data movement
  - Control
- Details in LC-3 ISA
  - Addressing modes
  - Condition codes
  - What the control unit does for each inst.

Important ideas from Chap. 6

- Problem solving
  - Step-wise refinement
  - Constructs: seq, cond, iter
- Debugging
  - What a debugger provides
  - How to debug: step, breakpoint, watchpoint
  - Types of mistakes: syntax, logic, data

Important ideas from Chap. 7

- Assembly language syntax
  - Instruction
  - Label
  - Comment
  - Pseudo-op
- Assembly process
  - Two-pass process
    - Symbol table
    - Location count

Important ideas from Chap. 8

- I/O (device) ←→ CPU
  - Device register
  - Memory-mapped vs. special instructions
  - Sync vs. async
  - Polling vs. interrupt

- LC-3 computer's keyboard & monitor I/O

Important ideas from Chap. 9

- The purpose of trap and subroutine

- Trap and subroutine
  - Trap instruction / JSR(R)
  - System control block
  - Return instruction
  - Register savings and restoring

Important ideas from Chap. 10

- Stack as an abstract data type
  - Push / pop, full / empty

- Interrupt-driven I/O
  - Program status register: priv, prio, cc
  - User vs. supervisor
  - Servicing & returning from interrupt

Important ideas from Chap. 11

- High-level vs. low-level language
- Interpretation vs. compilation

Important ideas from Chap. 12

- Variables
  - Data types: char, int, double
  - Scope: local, global
- Operators
  - Expression vs. statement
  - Precedence and associativity
- Variables and memory locations

Important ideas from Chap. 13

- Conditional constructs
  - If, if-else, switch
- Iterative constructs
  - While, for, do-while
- Other control structures
  - Break, continue

Important ideas from Chap. 14

- Functions
  - It's purpose
  - Prototype vs. definition
- Implementation a function
  - Activation record
  - Caller and callee interaction
  - Run-time stack

Important ideas from Chap. 15

- Debugging and testing philosophy

- Ways to reduce bugs and errors
  - Specification
  - Modularity
  - Defensive programming

Important ideas from Chap. 16

- Pointer
  - Address vs. value
  - Pointer as a variable
  - *: dereferencing
  - &: getting the address of
- Array
  - Array as a variable
  - Indexing arrays
  - Relation between pointers and arrays

Important ideas from Chap. 17

- Recursion
  - Recursion
    - Recursion
      - Recursion
      - Recursion
        - Recursion
          - Recursion
            - Recursion
              - Recursion
                - Recursion
                - Recursion
              - Recursion
                - Recursion
            - Recursion
              - Recursion
                - Recursion
        - Recursion
          - Recursion
            - Recursion
    - Recursion
      - Recursion
  - Recursion
  - Recursion
    - Recursion

Important ideas from Chap. 17

- Recursion vs. iteration
- Implementing a recursive function
  - Smallest problem (base-case)
  - Smaller problem (calling itself)
- How the runtime stack changes during recursive calls

Exercise 9.2

- How many trap service routines can be implemented in the LC-3?
- Why must be a RET instruction be used to return from a TRAP routine? Why won't a BR instruction work instead?
- How many accesses to memory are made during the processing of a TRAP instruction? Assume the TRAP is already in the IR.

Exercise 9.18

```
;;; should output “ABCFGH”
          .orig x3000
          LEA       R1, TESTOUT
BACK_1    LDR       R0, R1, #0
          BRz       NEXT_1
          TRAP      x21
          ----------------- (a)
          BRnzp     BACK_1
NEXT_1    LEA       R1, TESTOUT
BACK_2    LDR       R0, R1, #0
          BRz       NEXT_2
          JSR       SUB_1
          ADD       R1, R1, #1
          BRnzp     BACK_2
NEXT_2    ----------------- (b)
SUB_1     ----------------- (c)
K         LDI       R2, DSR
          ----------------- (d)
          STI       R0, DDR
          RET
DSR       .FILL     xFE04
DDR       .FILL     xFE06
TESTOUT   .STRINGz “ABC”
          .END
```

Exercise 10.8

```
PUSH A
PUSH B
POP
PUSH C
PUSH D
POP
PUSH E
POP
POP
PUSH F
```

```
PUSH G
PUSH H
PUSH I
PUSH J
POP
PUSH K
POP
POP
POP
PUSH L
POP
POP
PUSH M
```

Exercise 11.6

- The UNIX command line shell is an interpreter. Why can’t it be a compiler?

Exercise 12.5

- Write the LC-3 code that would result if the following local variable declarations were compiled using the LC-3 C compiler

```
char c = 'a';
int x = 3;
int y;
int z = 10;
```

Exercise 12.12

Say variables a and b are both declared locally as long int

- Translate the expression a+b into LC-3 code, assuming long int occupies two bytes. Assume a is allocated at offset 0 and b is at offset -1 in the activation record for their function.
- Translate the same expression, assuming a long int occupies four bytes, a is allocated offset 0, and b is at offset -2.

Exercise 12.20

Suppose a program contains two integer variables x and y, which have values 3 and 4, respectively. Write C statements that will exchange the values in x and y so that after the statements are executed, x is equal to 4 and y is equal to 3.

- First, write this routine using a temp variable.
- Now rewrite this routine without using a temp variable.

Exercise 14.15

```
/*
JSR inst to f @ x3102
JSR inst to g @ x3301
JSR inst to h @ x3304
User provided 4 5 6 as input
Draw runtime stack when about to return from f
*/

#include <stdio.h>
int f(int x, int y, int z);
int g(int arg);
int h(int arg1, int arg2);

int main(){
  int a, b, c;
  printf("Type three numbers: ");
  scanf("%d %d %d", &a, &b, &c);
  printf("%d", f(a, b, c));
}
```

```
int f(int x, int y, int z){
  int x1;
  x1 = g(x);
  return h(y, z) * x1;
}
int g(int arg){
  return arg * arg;
}
int h(int arg1, int arg2){
  return arg1 / arg2;
}
```

Exercise 15.3

```
// Identify and suggest ways to fix the error
# include <stdio.h>
int main(){
        int smallestNumber = 0;
        int nextInput;

        /* Get the first input number */
        scanf(“%d”, &nextInput);

        /* Keep reading inputs until user enters -1 */
        while (nextInput !=-1){
                if (nextInput < smallestNumber)
                        smallestNumber = nextInput;
                scanf(“%d”, &nextInput);
        }
        printf(“The smallest number is %d\n”, smallestNumber);
}
```

Exercise 15.7

```
// Identify the serious design errors; propose and implement a correct solution
#include <stdio.h>
#define SEATS 10
int main(){
  int seatsAvailable = SEATS;
  char request = ‘0’;
  while (request!=‘X’){
    scanf(“%c”, &request);
    if (request==‘R’){
      if (seatsAvailable) printf(“Reservation approved!\n”);
      else printf(“Sorry, flight fully booked.\n”);
    }
    if (request==‘P’){
      seatsAvailable -= 1;
      printf(“Ticket purchased!\n”);
    }
  }
  printf(“Done! %d seats not sold\n”, seatsAvailable);
}
```

Exercise 16.7

```
// What does this program perform by drawing out a run-time stack
# include <stdio.h>
int main(){
  int apple;
  int *ptr;
  int **ind;
  ind = &ptr;
  *ind = &apple;
  **ind = 123;

  ind++;
  *ptr++; // same as *(ptr++);
  apple++;

  printf("%x %x %d\n", ind, ptr, apple);
}
```

Exercise 16.10

- Write a program to find the median of a set of numbers.

Exercise 16.12

```
// Find the error
#include <stdio.h>
#define MAX_LEN 10
Char *LowerCase(char *s)
int main(){
        char str[MAX_LEN];
        printf(“Enter a string : “);
        scanf(“%s”, str);
        printf(“Lowercase: %s \n”, LowerCase(str));
}
char *LowerCase(char *s){
        char newStr[MAX_LEN];
        int index;
        for (index=0; index < MAX_LEN; index++){
                if (‘A’ <= s[index] && s[index] <= ‘Z’)
                        newStr[index] = s[index] + (‘a’ – ‘A’);
                else
                        newStr[index] = s[index];
        }
        return newStr;
}
```

Exercise 17.4

```
// what does count(20) return?
int count(int arg){
  if (arg<1) return 0;
  else if (arg%2) return 1+count(arg-2);
  else return 1+count(arg-1);
}
```