## Contents

Yiping Deng

September 14, 2018

## 1 Info

course website: `https://grader.eecs.jacobs-university.de/courses/320241/2018_2/`
office hours: Mondays 10:00 - 12:00
there are two TAs, they will hold tutorials.

### 1.1 books:

- digital systems
- computer organization and design
- Compilers: Principles, Techniques, and Tools(Dragon book)
- Compiler Design in C

### 1.2 goals:

understand the basic knowledge of:

- computer architecture
- data representation
- instruction set architectures
- datapath and control
  - why are logic gate important, how to optimize
- programming languages characteristics
- phases of compilation

### 1.3 grading scheme:

- 30% homework(in total 12)
- 30% midterm
- 40% final

### 1.4 homework:

- individual submission
- homework should be submitted in a single pdf, need a homework template
- homework is due on Tuesday 14:00 sharp

#### 1.4.1 grading criteria

- 10% homeowork formatting
- 50- 70% intermediate steps

### 1.5 tutorials:

- Sundays in West Hall 4 at 17:00

## 2 Introduction

### 2.1 history

1. mechanical machines
2. electronic mchines
3. digital computers
4. networking

### 2.2 processor

- Pentium I: 60MHz, 800 nm
- Nehalem: 45 nm
- Gulfdown: 32 nm

## 2.3 numerical representation and numerical system

### 2.3.1 analog v. digital

physical systems use quantities which must be manipulated arithmetically. Quantities may be represented numerically in either analog or digital form.

- analog: a continuous variable, proportional indicator
- digital: varies in discrete step

Table 1: Digital v. Analog

| Digital | Analog |
|---|---|
| discrete steps | a continuous variable |
| ease of design | |
| well suited for storing information | |
| accurate and easy to maintain | |
| programmable operation | |
| less affected by noise | |
| ease of fabrication | |

### 2.3.2 digital number systems

- decimal systems
  - 10 symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - each number is a digit
  - most significant digit (MSD) and least significant digit (LSD)
  - Positional value may be stated as a digit multiplied by a power of 10
- binary systems
  - lends itselfs to electronic circuit design
  - can convert to other representation
  - Positional value may be stated as a digit multiplied by a power of 2

#### 2.3.3 digital circuit v. logic circuit

- Digital circuits – produce and respond to predefined voltage ranges
- Logic circuits – used interchangeably with the term, digital circuits
- Digital integrated circuits (ICs) – provide logic operations in a small reliable package

#### 2.3.4 parallel v. serial

- Parallel transmission – all bits in a binary number are transmitted simultaneously. A separate line is required for each bit
- Serial transmission – each bit in a binary number is transmitted per some time interval
- Both methods have useful applications which will be seen in later chapters

#### 2.3.5 memory

- A circuit which retains a response to a momentary input is displaying memory
- Memory is important because it provides a way to store binary numbers temporarily or permanently
- Memory elements include:
  - Magnetic
  - Optical
  - Electronic latching circuits

# 3 Digital system

## 3.1 Conversion

### 3.1.1 Binary system

1. Binary to decimal conversion Just simply raise 2 to the corresponding power and time the digit from right to the left

2. Decimal to binary conversion Repeated integer division. We continuously divide the number by 2, note down all the reminder, and then reverse the order

### 3.1.2 Hexadecimal Number system

- base 16, 16 possible symbols, 0 - 9 and A - F
- uses groups of 4 bits
- allows for convenient handling of long binary strings

1. hexadecimal to decimal $163_{16} = 1 * (16^2) + 6 * (16^1) + 3 * (16^0)$
2. decimal to hexadecimal still using repeated integer division
3. hexadecimal to binary use the conversion table
4. binary to hexadecimal note that we group 4 bits from the right, fill in 0 on the left side if the number of digits are not divisiable by 4

### 3.1.3 Binary-coded decimal(BCD)

- each decimal digits are converted to binary accordingly.
- Programmable calculators manufactured by Texas Instruments.
- it is not a number system

### 3.1.4 Bytes, Nibbles and Words

- 1 byte = 8 bits
- 1 nibble = 4 bits
- 1 word = depending on a data bus width
  - x86: 32 bits
  - arm64: 64bits

### 3.1.5 ASCII code

- 0 - 127

## 3.2 Logic Functions

### 3.2.1 boolean constants and variables

- Boolean algebra allows only value 0 and 1
- Logic 0 can be: false / off / low / open switch
- Logic 1 can be: true / on / high / close switch
- three basic logic operators:
  - OR
  - AND
  - NOT

### 3.2.2 to generate truth table

```
from itertools import zip_longest as izip, product, tee

# Logic functions: take and return iterators of truth values

def AND(a, b):
    for p, q in izip(a, b):
        yield p and q

def OR(a, b):
    for p, q in izip(a, b):
        yield p or q

def NOT(a):
    for p in a:
        yield not p

def EQUIV(a, b):
    for p, q in izip(a, b):
        yield p is q

def IMPLIES(a, b):
    for p, q in izip(a, b):
        yield (not p) or q
```

```
def create(num=2):
    ''' Returns a list of all of the possible combinations of truth for the given numbe
    ex. [(T, T), (T, F), (F, T), (F, F)] for two variables '''
    return list(product([True, False], repeat=num))

def unpack(data):
    ''' Regroups the list returned by create() by variable, making it suitable for use
    ex. [(T, T, F, F), (T, F, T, F)] for two variables '''
    return [[elem[i] for elem in lst] for i, lst in enumerate(tee(data, len(data[0])))]

def print_result(data, result):
    ''' Prints the combinations returned by create() and the results returned by the lo
    n = len(data[0])
    headers = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'[:n]
    begin_format = '|'.join(n * 'c')
    begin_str = '\\begin{tabular}{|' + begin_format + '|c|}'
    print(begin_str)
    print('\\hline')
    print(' & '.join(headers) + ' & Result \\\\')
    print('\\hline')
    for row, result_cell in izip(data, result):
        print(' & '.join({True: 'T', False:'F'}[cell] for cell in row) + ' &' + ' ' + 
    print('\\hline')
    print('\\end{tabular}')
data = create(num=4)
A, B, C, D= unpack(data)
result = OR(OR(OR(A, B), C), D)
print_result(data, result)
```

## 3.3 Logic Circuits

### 3.3.1 Minimization of Logic Expression: rules

- $w(y+z) = wy + wz$ (distributive rule)
- $w + \overline{w} = 1$