CS230 course note

Chenxuan Wei

Jan 2022

Contents

1 Week 1

1.1 Number Representation

1. Base R number
   Assume given a n-digit number of base r which
   $x = d_{n\_1}d_{n-2}\dots d_1d_0$
   where $d = [0, r-1]$
   then $x = \sum_{i=0}^{n-1} d_i r^i$

2. Common base
   Decimal = base 10, Binary = base 2, Hexadecimal = base 16 (0x..... is a way to mark base 16)

3. Decimal $\rightarrow$ Other Base

   - Devide by the target base until quocient = 0
   - Get the remainder together backward

4. Base 2 $\iff$ Base 16
   group digits in group of 4

2 Week 2

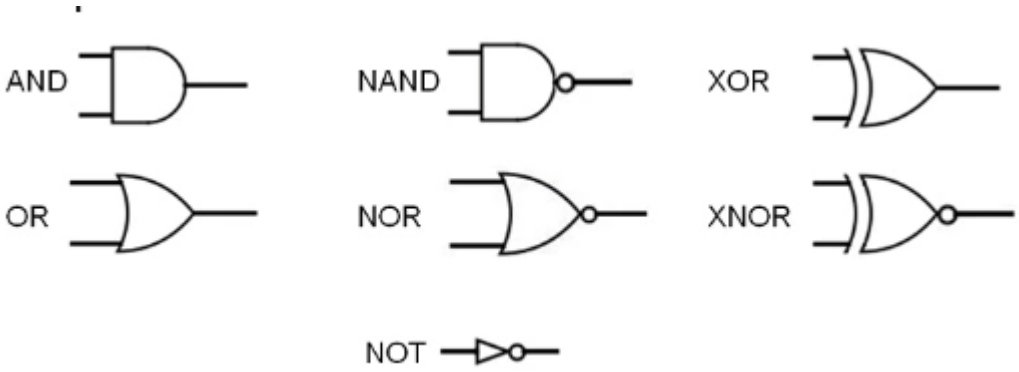
2.1 Boolean Algebra and circuits

1. different operator
   - or = $\cup$
   - and = $\cap$
   - not = $\neg$
   - NAND = |
   - NOR = a arrow down
   - XOR = $\oplus$ = exactly one
   - XNOR = a dot in a circle = can't be exactly one
2. Precedence
   $Not > and = NAND > XOR = XNOR > OR = NOR$

3. Algebra Rules

- Identities
  - $A \cup 0 = A$
  - $A \cap 1 = A$
  - $A \cup A = A$
  - $A \cap A = A$
- Involution
  $\neg\neg A = A$
- Anihilators
  - $A \cup 1 = 1$
  - $A \cap 0 = 0$
- Complements
  - $A \cup \neg A = 1$
  - $A \cap \neg A = 0$
- Commutaive Law
  - $A \cup B = B \cup A$
  - $A \cap B = B \cap A$
- Associative Law
  - $A \cup (B \cup C) = (A \cup B) \cup C$
  - $A \cap (B \cap C) = (A \cap B) \cap C$
- Distributive Law
  - $A \cap (B \cup C) = A \cap B \cup A \cap C$
  - $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
- De Morgan's Law
  - $\neg(A \cup B) = \neg A \cap \neg B$
  - $\neg(A \cap B) = \neg A \cup \neg B$

4. Digital Circuits



2.2 Binary Arithmetic and Two's Complement

1. Addition and multiplication
   just like normal stuff

2. Signed Magnitude
   MSB is sign

3. One complement
   Negative numbers are inverted positive numbers

4. Two's complement
   Negative numbers are inverted (each digit)positive number and add 1

3 Week 3

3.1 Floating Point Introduction

1. Formal fraction method: multiply fraction part by 2
   determin the number before the decimal place

2. From binary to decimal
   just do negative exponenets

3. Meaning of Point

   - Radix Point
     Decimal point for base-10
     Binary point for base -2
   - Floating point
     explicit exponents → radix point can move
     more flexible than fixed point
     wider range, variable precision
   - Fixed-point
     fixed number of digits before and after radix point
     less flexible, less range, but faster, more precision
     Can use integer arithmetic operations

4. Floarting point
   $(-1)^S * 1.F * 2^{E-B}$

   - 1 - integer
   - 2 - base
   - S - sign
   - F - fraction
   - E - exponent
   - B - bias
   - 1.F - significand or mantissa

5. Example:
   bit 0..3 fraction, bit 4...6 exponent bit 7 : sign, bias is 3

3.2 Floating Point continue

1. Normalized Representation Problem
   1.F with a leading one before decimal
   $(-1)^S * (1 + F) * 2^{E-B}$

2. Subnormal Representation
   $(-1)^S * (0 + F) * 2^{1-B}$
   If E is all zero, use subnormal
   if 2^e have $e < 1 - B$ must be subnormal

3. Special Cases

   - Overflow:
     E too large $\rightarrow$ represent as $+/-\infty$
     also for division by zero
   - Invalid result: NAN:
     $0/0$ or $\sqrt{-1}$
   - Both can be progragate

4. Special case formula

Exponent	Fraction	Case
00000	0000	0
00000	non-zero	subnormal
11111	00000	infinity
11111	non-zero	Nan
anything	anything	normal

5. Arithmetic

   - Addition
     align radix points
     use normal addition
   - Multipilication
     add exponents
     mutiply siginificnads

4 week 4

4.1 Endianness and Characters

1. Byte = 8 bits
   two's complement range [-128, 127]
   unsigned [0, 255]
   two hex digits

2. word = 32 or 64 bits
   Individual bytes still accessible
   the order are in

   - Little-endian: least-significan first
     same number in memory regradless of length
     can start math right away
   - Big-endian: most-significant first (in cs230)
     natural way of writing

   Converge it

   - converge to bit first
   - revert the bytes order

3. Characters
   ASCII characters - American Standard Code from hex to char
   Unicode - over 100000 code points

   - U+0000 - for unicode charactering mapping
   - UTF-8 - unicode transformation format
     1-4 bytes
     1 byte for ASCII

4. Bits have no inherent meaning
   must state the meaningat the begining

5. UTF-8 converging
   Check the leading of a byte

   - start with 0→ single byte character, ASCII character
   - start with 10 → continue byte
   - start with 1.10→ leading byte, show number of continue byte

4.2 Assembly Language Intro

1. Machine Code
   Binary - 0s and 1 s
   Direction execution by processor

2. Assembly Language
   Human-readable "programing language"
   Almost direct mapping to machine code

3. MIPS architecutre
   Microprocessor without Interlocked Pipeline Stages
   simplified version in CS230

4. Assembly
   takes 32 bits = 4 bytes = 1 word
   32 registers aviailable
   $0 = 0
   up to 3 register can be used
   1st destination, 2nd, 3rd are sources

5. Immediate Addition
   $addi\$t, \$s, i$
   is just $\$t = \$s + i$
   often use to initialize $addi\$t, \$0, i$

6. + and -
   $add\$d, \$s, \$t$
   is $\$d = \$s + \$t$
   $sub\$d, \$s, \$t$
   is $\$d = \$s - \$t$

7. * and /
   $mult\$s, \$t$
   lo = s*t
   $div\$s, \$t$
   lo is quotient, remainder in hi
   $mfhi\$d$ get stuff from hi
   $mflo\$d$ get stuff from lo

5 Week 5

5.1 Conditional Execution

1. beq $\$s, \$t, i$
   compare s and t, if equal, skip i instructions

2. bne $\$s, \$t, i$
   compare s and t, if not equal, skip i instructions

5.2 Emeory and Input/Output

1. $lw\$t, i(\$s)$
   let t = content in address in s+i

2. $sw\$t, i(\$s)$
   put content in t into s+i address

3. input/outpte
   0xFFFF0004 = input, 0xFFFF000C = output

6 Machine Internals

6.1 Basic control

1. Clock cycle
   beat of computer
2. Cycle Execution
   - One Instuction per clock cycle
     fixed cycle length must cover slowest intstruction
     which will slow down the speed
   - One instruction - multiple clock cycles
     each instruction is divided into subtasks
     some instructions take fewer cycles than others
     ex. lw take 5, add takes 4
3. Typical Instruction cycle
   - IF: instruction fetch
     load 32-bit intstruction from addresss in PC, pass it to ID
     Increment PC by 4
   - ID: instruction decode
     decode instruction
   - EX: execute
     run the instruction
   - MEM: memory access
     access memory
   - WB: write back
     write results back to registers
4. Performance
   - elapsed time
     total response time
   - CPU time
     time spent processing instruction
5. Clock period
   SI units of time in seconds per clock cycle
   $10^9 * 10^{-12} s = 10^9 ps = 10^6 ns = 10^3 \mu s = ms = 10^{-3}$
6. Clock frequency
   cycle persecond: Hertz(Hz)
   $10^{12} = Thz = 10^3 Ghz = 10^6 MHz = 10^9 Khz = 10^9 * 10^3$
   250ps = 4GHz

7. CPI
   = clock cycles / instruction

8. CPU time
   - Instruction count × CPI × Cycle time

6.2 pipelining

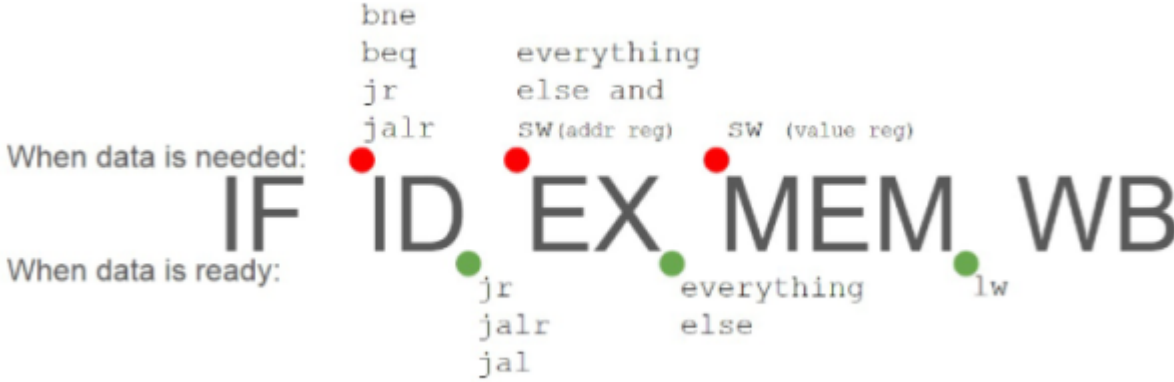
1. Speed up
   if all stages are balanced
   time between instruction(pipelined) = time between instructions (no pipelined) / number of stages
   if not balanced, speed up less

2. Pipeline diagram
   show what MIPS is doing for each clock cycles

3. Hazards

   - Structural: Combination of instruction types
     Resource is busy
   - Data: dependency between instruction
     need to wait for data read/write
     always 2 bubble
   - Control: dependency between instruction
     control dependes on previous instruction
     always 1 bubble

4. Forwarding
   allowed to use resource most efficiently

   Branch will be ready after ID

5. Branch prediction

   - Static branch prediction
     forward not taken, backward taken

- Dynamic branch prediction
  Depends on the result of last same branch (first one like static branch prediction)

6.3 Memory hierarchy

1. Tems
   - hit: data found in cache
   - miss: data not found, get it in main memory
   - hit time: cost of direct fetch
   - miss penalty: cost of get from main
   - byte-addressable: each bytye have it's own address
   - evict: the content in cache got kicked
2. Direct-mapped cache
   Assume M block of cache with size B
   request address p
   map to cache $c = (p/B) mod M$
3. Fully associative caches
4. 2-way associative
   Combine 2 cache, allowed more space in each
   i.e. have 00, 01, 10, 11 for direct, but 2 wayassoiciative will combine by 2, remain 0 (contain 00, 10), 1(contain 01, 11)
5. full associative
   combine all together, size = B
6. CPI calculation
   - Instruction cache miss rate * miss penalty + Base CPI + % of memory access of instruction * data cache miss eate * miss penalty
7. Average memoty access time
   = hit time + miss rate * miss penalty

7 Build and Execute

1. Deterministic Finite AUtomata(DFA)
   - Have exactly one start state
   - Have at least one final state
   - Finite set of input alphabet
   - One transition per alphabet per state
2. Non-deterministic finite automata (NFA)
   - Can have many transition per alphabet per state
   - can have ϵ symbol, go to new sate without consuming input
   - all DFA is NFA
   - can always get DFA from NFA
3. Regular expression
   - $R|S$ is means one of it
   - RS means S after R
   - $R*$ means $0 - \infty$ number of R
   - $a+$means $1 - \infty$ number of a
   - $a?$ means $0 - 1$ number of a
   - . represent anything
   - $[a - z]$ means anything between them
4. UNIX tools
   - egrep: search regular expression in text files
   - sed: stream editor for transforming text files
   - awk: pattern scanning and processing language
   - make: software building utlility
5. Scanner
   convert input string to tokens, use regular languages
6. Specification Components
   Terminal/token: atomic symbol
   no-terminal/variable: abstract component
   - does not literally appear in input
   - designated start symbol
   - notation: angle brackets

Rules: go from one variable to another
ϵ: empty, can be a terminal

7. Derivation
   application of rules to generate valid input string
   replace one variable by one rule
   continue until everything is token
8. Context-free grammar
   - formalism to specify languages
   - context-free: no overlap between blocks
9. Leftmost Derivation
   always start form the leftmost non-terminal
10. Prase trees (derivation trees)
    non-terminal are internal nodes
    terminals are leaf nodes
11. Associativity and precedence
    order of evaluating expression:
    associativity: gourping equivalanet operations
    precedence: grouping non-equivalent symbols
12. Tools
    - lex: scanner generator
      degine regular expression rules
      generates scanner
    - yaccL parser generator
      define context-free grammar
      generates parser
13. Asswembler
    converge one line instruction to one line machines code
14. Linking
    - Goal: combine multiple boject files
      avoid compiling whole program each time
    - resolve external symbols
    - produce executable file

8 Concurrency

1. Deadlocks
   if 2 threads contain a lock that each other wants, and they are waiting each other, they will wait forever
   which is deadlock