香港中文大學
The Chinese University of Hong Kong

CSCI2510 Computer Organization

Tutorial 05: Review for Midterm

Bentian Jiang

btjiang@cse.cuhk.edu.hk

Midterm Exam Announcement

- The midterm exam will be conducted on Oct. 23 (Tue) (scope: Lec01 ~ Lec05, HW01~HW02).

- The contents of “Tutorial 04: Stack and Queue” will NOT be included in the midterm exam.

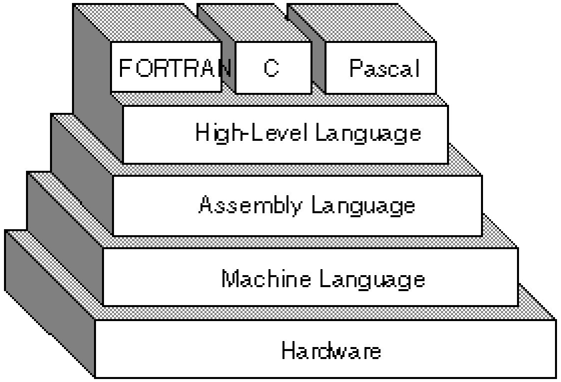
- Please also don’t worry about the programming exercise 2 (stack and queue), TA will give you more materials and hints in the next tutorial on Oct. 23.
  - The deadline for programming exercise 2 is Oct. 30.

Outline

- Assignment 1 Solution
- Assignment 2 Hint
- (Optional) Bit-wise Instruction Basic
  - They are important knowledge in CS area.
  - The contents will not be included in neither midterm nor final examinations.

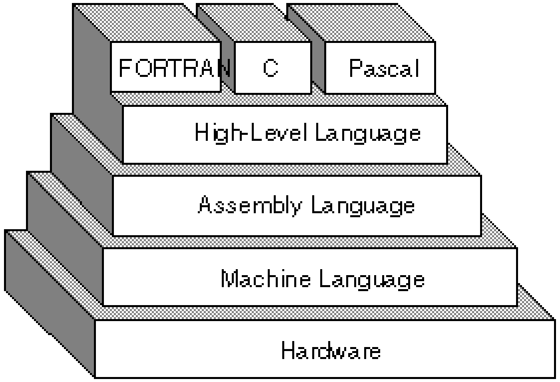
Assignment 1 Solution

- Q1 (1):
  - Cache Memory:
    - A smaller, faster RAM to hold parts of a program (and data) that are currently being executed by CPU.
  - Primary Memory:
    - A fast memory that operates at electronic speeds. Secondary Storage: Additional, less expensive, permanent secondary storage is used when large amounts of data and many programs have to be stored.
- Q1 (2):



Assignment 1 Solution

- Q1 (2):
  - As shown in figure, high-level languages (like C/C++) are designed to make the programming task easier by providing a more humanly understandable syntax, they should be compiled or interpreted to a low level machine language so a machine can finally execute. A high-level language will be translated to assembly language instructions and further interpreted into executable machine language code.

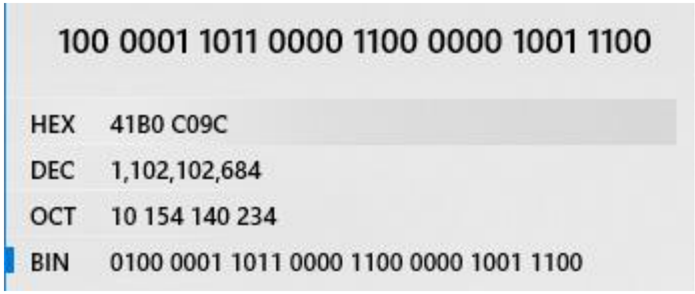


Assignment 1 Solution

- Q2 (1): BE4F3F64h
  - ¥O?d
  - Just translate it directly according to Hex in ASCII table
- Q2 (2):
  - BE4F3F64h
  - Unsigned Binary (1_h -> 4_b):
    - 1011 1110 0100 1111 0011 1111 0110 0100

Assignment 1 Solution

- Q2 (3):
  - **Signed** integer 2's-complement: (2's = 1's + 1)
    - 1100 0001 1011 0000 1100 0000 1001 1100



    - -1102102684
- Q2 (4):
  - **Signed** integer 1's-complement:
    - 1100 0001 1011 0000 1100 0000 1001 1011
    - -1102102683

Assignment 1 Solution

- Q2 (5):
  - signed integer using sign-and-magnitude
  - 1011 1110 0100 1111 0011 1111 0110 0100
  - -1045380964

- Q3 (1):
  - 8GB = 2^3 x 2^3 x 2^10 x 2^10 x 2^10 = 2^36 bits
  - 8, Byte to bit, KB to Byte, MB to KB, GB to MB
  - 2^36 bits, 2^33 bytes, 2^31 words (for four-byte word) or 2^32 words (for two-byte word)
  - Be careful about bit and byte!

Assignment 1 Solution

- Q3 (2):
  - Notice by (1), the memory system has 2^33 bytes. Hence, in order to represent the 2^33 bytes uniquely, the address should at least contains 33 bits.

- Q3 (3):
  - 3B12AA27h

Location	100	101	102	103
Little endian	27h	AAh	12h	3Bh
Big endian	3Bh	12h	AAh	27h

Assignment 2 Hint

- Basic Concepts of four common condition flags:

N (negative)	Set to 1 if the result is negative; otherwise, cleared to 0
Z (zero)	Set to 1 if the result is 0; otherwise; otherwise, cleared to 0
V (overflow)	Set to 1 if arithmetic overflow occurs; otherwise, cleared to 0
C (carry)	Set to 1 if a carry-out occurs; otherwise, cleared to 0

Assignment 2 Hint

- Given two 4-bit registers R1 and R2 storing signed integers in 2's-complement format. Please specify the condition flags that will be affected by **Add R2, R1**:
  - 1) R1 = $(7)_{10} = (0111)_2$,   R2 = $(3)_{10} = (0011)_2$
    - N = 1
    - Z = 0
    - V = 1
    - C = 0
  - 2) R1 = $(7)_{10} = (0111)_2$,   R2 = $(-5)_{10} = (1011)_2$
    - N = 0
    - Z = 0
    - V = 0
    - C = 1

Assignment 2 Hint

- Basic concepts of addressing modes:

Address Mode	Assembler Syntax	Addressing Function
1) Immediate	$\#Value$	$Operand = Value$
2) Register	Ri	$EA = Ri$
3) Absolute	LOC	$EA = LOC$
4) Register indirect	(Ri)	$EA = [Ri]$
5) Index	$X(Ri)$	$EA = [Ri] + X$
6) Base with index	(Ri, Rj)	$EA = [Ri] + [Rj]$

  – EA: effective address; Value: a signed number; X: index value

Assignment 2 Hint

- Determine the effective address (EA) of the last operand
  - ADD R1, R2
    - EA = R2
  - LOAD R1, (R2, R3)
    - EA = ?
  - MOV R1, LOC
    - EA = ?
  - LOAD R1, -C (R2)
    - EA = ?

Bit-wise Instruction Basic

- Bitwise Logic Instructions
  - NOT, AND, OR, XOR
  - For each bit:
    - NOT outputs 1 only if the input is 0
    - AND outputs 1 only if both inputs are 1
    - OR outputs 1 if at least one input is 1
    - XOR outputs 1 if exactly one input is 1
  - In C
    - NOT: a = ~ b;
    - AND: a = a & b;
    - OR: a = a | b;
    - XOR: a = a ^ b;

Bit-wise Instruction Basic

- Note that ANDing a bit with 0 produces a 0 at the output while ANDing a bit with 1 produces the original bit. This can be used to create a mask.
  - If you want to reserve the last 2 hex digits:
    - 1234h AND 00ffh
    - 0001 0010 0011 0100 AND 0000 0000 1111 1111
    - 0000 0000 0011 0100 = 0034h

- Question: “2” -> 2, how to convert ASCII ‘2’ (32H) to a byte with the value of 2?

Bit-wise Instruction Basic

- Similarly, note that ORing a bit with 1 produces a 1 at the output while ORing a bit with 0 produces the original bit. This can be used to force certain bits of a string to 1s.
  - 1234h OR 00ffh
  - 0001 0010 0011 0100 OR 0000 0000 1111 1111
  - 0001 0010 1111 1111 = 12ffh

- Question: 2 -> “2”: how to convert a byte with the value of 2 to ASCII ‘2’ (32H)?

Bit-wise Instruction Basic

- Additionally, note that XORing a bit with 1 produces flips the bit (0 -> 1, 1 -> 0) at the output while XORing a bit with 0 produces the original bit.

- It tells **whether two bits are unequal**.

- It is an **optional bit-flipper** (the deciding input chooses whether to invert the data input). How to use XOR to flip all the bits (i.e., NOT)?

- Question: How to initialize a register (clear the content in register) using a simple instruction?

Stack and Queue

- Stack and queue are very basic and important structures! There many algorithms and data structures are implemented base on stack and queue
  - E.g., how can we use stack?
  - A basic algorithm question which often appears in interview:
    - How to check for balanced parentheses in an expression?
    - E.g. exp = “[()]{}{[()()]()}”
    - Expected:
      - Time Complexity: O(n)
      - Space: O(n)

Summary

- Assignment 1 Solution
- Assignment 2 Hint
- Bit-wise Instruction Basic