CPSC-240 Computer Organization and Assembly Language

Chapter 10

Program Development

Instructor: Yitsen Ku, Ph.D.
Department of Computer Science,
California State University, Fullerton, USA

CALIFORNIA STATE UNIVERSITY
FULLERTON™

Outline

- Understand the Problem
- Create the Algorithm
- Implement the Program
- Test/Debug the Program
- Error Terminology
  - Assembler Error
  - Run-time Error
  - Logic Error

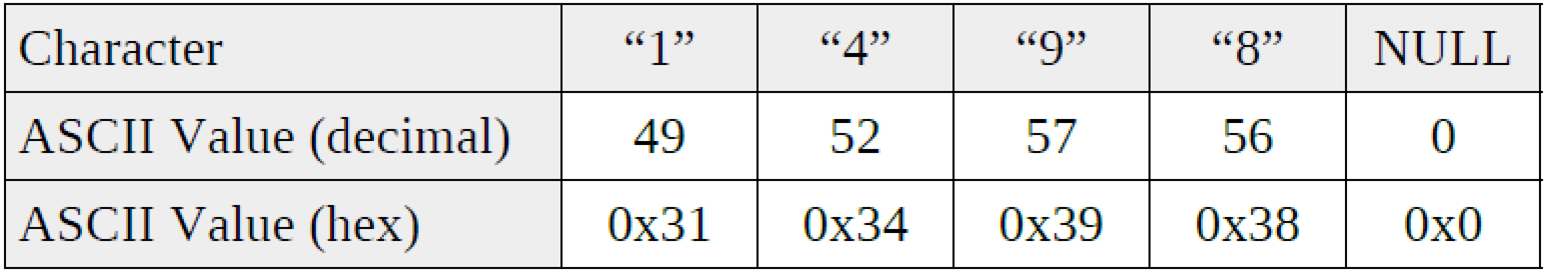
Understand the Problem

Understand the Problem

- The first step is to understand what is required, especially the applicable input information and expected results or output.
- An integer can be used for numeric calculations, but cannot be displayed to the console (as it is).
- A string can be displayed to the console but not used in numeric calculations.

For Example

- As an unsigned double-word integer, the numeric value 149810 would be represented as 0x000005DA in hex (double-word sized). The integer number 149810 (0x000005DA) would be represented by the string “1”, “4”, “9”, “8” with a NULL termination as follows.



Character	“1”	“4”	“9”	“8”	NULL
ASCII Value (decimal)	49	52	57	56	0
ASCII Value (hex)	0x31	0x34	0x39	0x38	0x0

Create the Algorithm

Create the Algorithm

- The process for creating an algorithm can be different for different people. In general, some time should be devoted to thinking about possible solutions.
- This may involve working on some possible solutions using a scratch piece of paper. Once an approach is selected, that solution can be developed into an algorithm.
- The algorithm should be written down, reviewed, and refined. The algorithm is then used as the outline of the program.

Integer to ASCII Conversion

Single Digit Integer to ASCII Conversion

- To convert a single digit integer (0-9) into a character, 4810 (or “0” or 0x30) can be added to the integer. For example, 0x01 + 0x30 is 0x31 which is the ASCII value of “1”.

Integer	Conversion	ASCII
0 = 0x00	0x00 + 0x30	0x30 = '0'
1 = 0x01	0x01 + 0x30	0x31 = '1'
2 = 0x02	0x02 + 0x30	0x32 = '2'
3 = 0x03	0x03 + 0x30	0x33 = '3'
4 = 0x04	0x04 + 0x30	0x34 = '4'
5 = 0x05	0x05 + 0x30	0x35 = '5'
6 = 0x06	0x06 + 0x30	0x36 = '6'
7 = 0x07	0x07 + 0x30	0x37 = '7'
8 = 0x08	0x08 + 0x30	0x38 = '8'
9 = 0x09	0x09 + 0x30	0x39 = '9'

A Larger Integer to a String Conversion

- In order to convert a larger integer (≥10) into a string, the integer must be broken into its component digits.
- For example, 123_{10} (0x7B) would be 1, 2, and 3. This can be accomplished by repeatedly performing integer division by 10 until a 0 result is obtained.

$$\frac{123}{10} = 12 \;\; remainder\ 3$$

$$\frac{12}{10} = 1 \quad remainder\ 2$$

$$\frac{1}{10} = 0 \quad remainder\ 1$$

Convert A Larger Integer to a String Algorithm

```
int intNum = 1498;
char strNum[10];
register int rcx = 0, rdi = 0;
do {
        push intNum%10;
        rcx++;
} while(intNum/10 != 0);
do {
        pop rax;
        rax += 0x30;
        strNum[rdi++] = al;     // strNum[rdi] = al; rdi++;
} while(rcx-- > 0);             // while(rcx>0); rcx--;
```

Implement the Program

Implement the Program (1)

```
; Simple example program to convert an
; integer into an ASCII string.
; ********************************************************
; Data declarations
section .data
; -----
; Define constants
NULL      equ    0
EXIT_SUCCESS equ 0                ; successful operation
SYS_exit  equ    60               ; code for terminate
; -----
; Define Data.
intNum    dd     1498
section .bss
strNum    resb   10
; ********************************************************
```

Implement the Program (2)

```
section .text
global _start
_start:
; Convert an integer to an ASCII string.
; -----
; Part A - Successive division
        mov     eax, dword [intNum]     ; get integer
        mov     rcx, 0                  ; digitCount = 0
        mov     ebx, 10                 ; set for dividing by 10
divideLoop:
        mov     edx, 0
        div     ebx                     ; divide number by 10
        push    rdx                     ; push remainder
        inc     rcx                     ; increment digitCount
        cmp     eax, 0                  ; if (result > 0)
        jne     divideLoop              ; goto divideLoop
```

Implement the Program (3)

```
; -----
; Part B - Convert remainders and store
          mov       rbx, strNum                   ; get addr of string
          mov       rdi, 0                        ; idx = 0
popLoop:
          pop       rax                           ; pop intDigit
          add       al, "0"                       ; char = int + "0"
          mov       byte [rbx+rdi], al            ; string[idx] = char
          inc       rdi                           ; increment idx
          loop      popLoop                       ; if (digitCount > 0)
; goto popLoop
          mov       byte [rbx+rdi], NULL          ; string[idx] = NULL
; -----
; Done, terminate program.
last:
          mov       rax, SYS_exit                 ; call code for exit
          mov       rdi, EXIT_SUCCESS             ; exit with success
          syscall
```

ASCII to Integer Conversion

Single Character to Integer Conversion

- To convert a single character ('0' ~ '9') into an integer, '0' (or 0x30 or 48) can be subtracted to the character. For example, the ASCII of '5' is 0x35, and 0x35 - 0x30 = 0x05 (or 5).

ASCII	Conversion	Integer
'0' = 0x30	0x30 - 0x30	0x00 = 0
'1' = 0x31	0x31 - 0x30	0x01 = 1
'2' = 0x32	0x32 - 0x30	0x02 = 2
'3' = 0x33	0x33 - 0x30	0x03 = 3
'4' = 0x34	0x34 - 0x30	0x04 = 4
'5' = 0x35	0x35 - 0x30	0x05 = 5
'6' = 0x36	0x36 - 0x30	0x06 = 6
'7' = 0x37	0x37 - 0x30	0x07 = 7
'8' = 0x38	0x38 - 0x30	0x08 = 8
'9' = 0x39	0x39 - 0x30	0x09 = 9

A String to a Larger Integer Conversion

- In order to convert a string into a larger integer (≥10), each character of the string must first be converted to an integer and then combined to form a large integer.
- For example, to convert '123' into 123, the string '123' = 0x31, 0x32, 0x33 should be subtracted by 10 to get 0x01, 0x02, 0x03. After that, each number is multiplied by their weight, such as 1*100, 2*10, and 3*1, and then these three numbers are added to get the final value of 123.

A String to a Larger Integer Conversion

- First step: convert each character into a integer

Address	ASCII+0	ASCII+1	ASCII+2
Character	'1' = 0x31	'2' = 0x32	'3' = 0x33
Subtract	0x30	0x30	0x30
Integer	1	2	3

- Second step: each number is multiplied by their weight 1*100, 2*10, and 3*1.
- Third step: the three numbers are added to get the final value
- $1 \times 100 + 2 \times 10 + 3 = (1 \times 10 + 2) \times 10 + 3 = 123$

Convert a String to a Larger Integer Algorithm

```
short shortNum;
char strNum[4] = "1234";
int strLen = 4
register int rsi = 0, rdi = 10;
do {
        strNum[rsi] &= 0x0f;
        al += strNum[rsi];
        if(rsi < strLen-2) {
                al *= 10;
        }
} while(rsi++ < strLen-1);
shortNum = al;
```

Implement the Program

Implement the Program (1)

```
; Convert an ASCII into an Integer.
; ********************************************************
; Data declarations
section .data
; -----
; Define constants
NULL      equ    0
EXIT_SUCCESS equ 0                ; successful operation
SYS_exit  equ    60               ; code for terminate
; -----
; Define Data
strNum    db     "1498"
section .bss
intNum    resw   1
; ********************************************************
section .text
          global _start
_start:
          mov    rax, 0            ;clear rax
          mov    rdi, 10           ;rdi = 10
          mov    rbx, strNum       ;rbx = address of strNum
          mov    rsi, 0            ;counter = rsi = 0
```

Implement the Program (2)

```
; -----
; Convert an ASCII string to an Integer.
next:
       and    byte[rbx+rsi], 0fh  ;convert strNum to number
       add    al, byte[rbx+rsi]   ;al = number
       adc    ah, 0               ;ah = 0
       cmp    rsi, 3              ;compare rsi with 3
       je     skip                ;if rsi=3 goto skip
       mul    di                  ;dx:ax = ax * di
skip:
       inc    rsi                 ;rsi++
       cmp    rsi, 4              ;compare rsi with 4
       jl     next                ;if rsi<=3 goto next
       mov    word[intNum], ax    ;intNum = ax
; -----
; Done, terminate program.
last:
       mov    rax, SYS_exit       ; call code for exit
       mov    rdi, EXIT_SUCCESS   ; exit with success
       syscall
```

Test/Debug the Program

Display a String in ddd Debugger

- **x/s &strNum** display the string address and the contents:

```
(gdb) x/s &strNum
0x600104: "1498"
```

- **x/5cb &strNum** show the address of the string followed by both the decimal and ASCII representation:

```
(gdb) x/5cb &strNum
0x600104: 49 '1' 52 '4' 57 '9' 56 '8' 0 '\000'
```

Error Terminology

Assembler Error

- Assembler errors (syntax errors) are generated when the program is assembled.
- The assembler will provide a list of errors and the line number of each error. It is recommended to address the errors from the top down. Resolving an error at the top can clear multiple errors further down.
- Typical assembler errors include misspelling an instruction and/or omitting a variable declaration.

Run-time Error

- A run-time error is something that causes the program to crash.

Logic Error

- A logic error is when the program executes, but does not produce the correct result.
- For example, coding a provided formula incorrectly or attempting to compute the average of a series of numbers before calculating the sum.

End of Chapter 10