Белгородский Государственный Технологический Университет им. В. Г. Шухова

Кафедра программного обеспечения вычислительной техники  
и автоматизированных систем

## Лабораторная работа №7 «Команды поразрядной обработки данных»

Выполнил:  
студент группы ПВ-31  
Адаменко И. И.

Проверил:  
Осипов О. В.

Белгород  
2014

### Вариант 1

Цель работы: изучить команды поразрядной обработки данных и особенности их применения.

### Задания к работе

1. Ознакомиться с теоретическим материалом.
2. Исследовать выполнение всех команд поразрядной обработки данных с любыми возможными типами операндов.
3. Решить поставленные задачи с помощью MASM.
4. В отчёт включить все необходимые листинги.

### Индивидуальное задание

1. В слове установить 0, 2 и 5 биты, сбросить 7, 11 и 13 биты, инвертировать 3, 8 и 15 биты. Остальные биты оставить без изменения.
2. В слове обнулить крайний справа единичный бит.

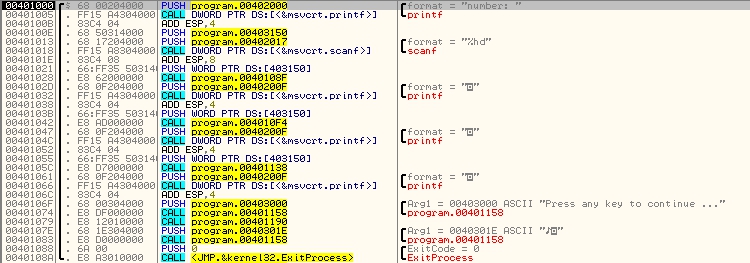
Отчёт

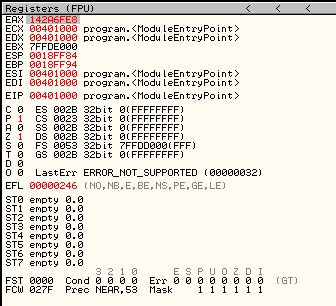
Напишем программу для MASM, решающую поставленную задачу. Пусть значения переменных вводятся с клавиатуры.

Код программы:

|  |  |
| --- | --- |
| 001 | include \masm32\include\masm32rt.inc |
| 002 |  |
| 003 | .const |
| 004 | EnterN db "number: ", 0 |
| 005 | InBin db "bin: ", 0 |
| 006 |  |
| 007 | NewLine db 10, 0 |
| 008 |  |
| 009 | One db "1", 0 |
| 010 | Zero db "0", 0 |
| 011 | Space db " ", 0 |
| 012 |  |
| 013 | InputFormat db "%hd", 0 |
| 014 | OutputFormat db "result: %d", 0 |
| 015 |  |
| 016 | .data |
| 017 |  |
| 018 | .data? |
| 019 | n dw ? |
| 020 |  |
| 021 | .code |
| 022 |  |
| 023 | inputing macro tip, format, variable |
| 024 | invoke crt\_printf, addr tip |
| 025 | invoke crt\_scanf, addr format, addr variable |
| 026 | endm |
| 027 |  |
| 028 | printing macro format, variable |
| 029 | invoke crt\_printf, addr format, variable |
| 030 | endm |
| 031 |  |
| 032 | printStr macro string |
| 033 | invoke crt\_printf, addr string |
| 034 | endm |
| 035 |  |
| 036 | main proc |
| 037 |  |
| 038 | inputing EnterN, InputFormat, n |
| 039 |  |
| 040 | PUSH n |
| 041 | call printBin |
| 042 | printStr NewLine |
| 043 |  |
| 044 | PUSH n |
| 045 | call first |
| 046 | printStr NewLine |
| 047 |  |
| 048 | PUSH n |
| 049 | call second |
| 050 | printStr NewLine |
| 051 |  |
| 052 | Inkey |
| 053 | invoke ExitProcess, NULL |
| 054 |  |
| 055 | main endp |
| 056 |  |
| 057 | printBin proc |
| 058 |  |
| 059 | printStr InBin |
| 060 |  |
| 061 | POP EBX ; save return address |
| 062 | MOV ECX, 16 ; ECX = 16, number of bits in word |
| 063 | MOV ESI, 0 ; printed digits = 0 |
| 064 | MOV EDI, 4 ; digits in one part = 4 |
| 065 |  |
| 066 | cycle: |
| 067 | POP DX ; get n |
| 068 | ROL DX, 1 ; cyclic shift DX to left; left bit to CF |
| 069 | PUSH DX ; save n |
| 070 | PUSH ECX ; save ECX |
| 071 |  |
| 072 | JB printOne ; if CF == 1 then print `1` |
| 073 |  |
| 074 | printStr Zero ; print `0` |
| 075 | JMP endcycle ; to end of loop |
| 076 |  |
| 077 | printOne: |
| 078 | printStr One ; print `1` |
| 079 |  |
| 080 | endcycle: |
| 081 | INC ESI ; printed digits += 1 |
| 082 | CMP ESI, EDI ; ESI == EDI ? ZF = 1 |
| 083 | JE printSpace ; if ESI == EDI then print ` ` |
| 084 | JMP endd |
| 085 |  |
| 086 | printSpace: |
| 087 | MOV ESI, 0 ; printed digits = 0 |
| 088 | printStr Space ; print ` ` |
| 089 |  |
| 090 | endd: |
| 091 | POP ECX ; restore ECX |
| 092 | LOOP cycle |
| 093 |  |
| 094 | PUSH EBX ; restore return address |
| 095 | ret |
| 096 |  |
| 097 | printBin endp |
| 098 |  |
| 099 | first proc |
| 100 |  |
| 101 | POP EBX ; save return address |
| 102 | POP AX ; n |
| 103 |  |
| 104 | ; for example: n = 1000, n\_2 = 0000 0011 1110 1000 |
| 105 | BTS AX, 0 ; n\_2 = 0000 0011 1110 100[1] |
| 106 | BTS AX, 2 ; n\_2 = 0000 0011 1110 1[1]01 |
| 107 | BTS AX, 5 ; n\_2 = 0000 0011 11[1]0 1101 |
| 108 | BTR AX, 7 ; n\_2 = 0000 0011 [0]110 1101 |
| 109 | BTR AX, 11 ; n\_2 = 0000 [0]011 0110 1101 |
| 110 | BTR AX, 13 ; n\_2 = 00[0]0 0011 0110 1101 |
| 111 | BTC AX, 3 ; n\_2 = 0000 0011 0110 [0]101 |
| 112 | BTC AX, 8 ; n\_2 = 0000 001[0] 0110 0101 |
| 113 | BTC AX, 15 ; n\_2 = [1]000 0010 0110 0101 |
| 114 | ; result: n\_2 = 1000 0010 0110 0101, n = 33381 |
| 115 |  |
| 116 | printing OutputFormat, AX |
| 117 |  |
| 118 | endd: |
| 119 | PUSH EBX ; restore return address |
| 120 | ret |
| 121 |  |
| 122 | first endp |
| 123 |  |
| 124 | second proc |
| 125 |  |
| 126 | POP EBX ; save return address |
| 127 | POP AX ; n |
| 128 |  |
| 129 | ; for example: n = 1000, n\_2 = 0000 0011 1110 1000 |
| 130 | BSF DX, AX ; n\_2 = 0000 0011 1110 1000 |
| 131 | ; ^ third bit equals 1 |
| 132 | ; => DX = 3 |
| 133 | BTR AX, DX ; n\_2 = 0000 0011 1110 [0]000 |
| 134 | ; result: n\_2 = 0000 0011 1110 1000, n = 992 |
| 135 |  |
| 136 | printing OutputFormat, AX |
| 137 |  |
| 138 | endd: |
| 139 | PUSH EBX ; restore return address |
| 140 | ret |
| 141 |  |
| 142 | second endp |
| 143 |  |
| 144 | end main |

Скриншоты программы из OllyDbg:





Результат работы программы:

