**Міністерство освіти і науки України**

**Харківський Радіотехнічний технікум**

**Циклова комісія ЕОТ**

**Адміністративна контрольна робота**

**з навчальної дисципліни**

**“Програмування мовою асемблер ”**

**зі спеціальності 5.05010201**

**“ Обслуговування комп’ютерних систем і мереж”**

**Варіант № 1**

**Виконав студент групи ОТ-335 Акулов Д.О.**

**м. Харків 2017-2018**

Для **Варіанту № 1** операційна підтримка має таку структуру

**Begin**

**While Y < X do**

**Y:= Y + 1;**

**X:= X or Y ;**

**Repeat**

**X:= X - 1**

**Until X <= Y**

**End.**

Спрощена схема алгоритму будується згідно з операційною підтримкою

**Begin**

**Y - X**

**Y< X**

**Y := Y + 1**

**X or Y**

**X <Y**

**X - Y**

**X := X - 1**

**End**

**0**

**0**

**1**

**1**

**Рисунок 1 – Спрощена схема алгоритму реалізації в комп’ютері програмних циклів**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Таблиця 1 – Стани ланцюга змінних до програмних циклів, після їх**  **виконання з урахуванням зміни стану після дії заданої функції** | | | | | | | | |
| **Initial state** | | | | | | | | |
| **X** | **0\_00** | **126\_7E** | **64\_40** | **1\_01** | **255\_FF** | **193\_C1** | **64\_40** | **192\_C0** |
| **Y** | **0\_00** | **64\_40** | **126\_7E** | **0\_00** | **128\_80** | **127\_7F** | **248\_F8** | **15\_0F** |
| **While Y < X do Y:= Y - 1** | | | | | | | | |
| **X** | **0\_00** | **126\_7E** | **64\_40** | **1\_01** | **255\_FF** | **193\_C1** | **64\_40** | **192\_C0** |
| **Y** | **1\_01** | **127\_7F** | **126\_7E** | **2\_02** | **0\_00** | **127\_7F** | **65\_0F** | **15\_0F** |
| **X:= X xor Y** | | | | | | | | |
| **X** | **1\_01** | **127\_7F** | **126\_7E** | **3\_03** | **255\_FF** | **255\_FF** | **65\_41** | **207\_CF** |
| **Y** | **1\_01** | **127\_7F** | **126\_7E** | **2\_02** | **0\_00** | **127\_7F** | **65\_41** | **15\_0F** |
| **Repeat X:= X + 1 Until X >= Y** | | | | | | | | |
| **X** | **0\_00** | **126\_7E** | **125\_7D** | **1\_01** | **254\_FE** | **245\_FE** | **64\_40** | **206\_CE** |
| **Y** | **1\_01** | **127\_7F** | **126\_7E** | **2\_02** | **0\_00** | **127\_7F** | **65\_41** | **15\_0F** |

Будується таблиця станів в пам’яті ланцюгів операндів до реалізації в комп’ютері програмних циклів, після послідовного їх виконання, включаючи реалізацію арифметико логічної функції. Стани в Таблиці 1 представлені в десятковій та шістьнадцятковій системах числення.

Детальна схема алгоритму будується по спрощеній та з урахуванням модульності її структури. Мається на увазі використання створених програмних макросів та викликів процедур. Приведена детальна схема алгоритму на рисунку унизу має лінійний характер, але на певних ділянках викликаються макроси та процедури, які по своїй структурі є циклічними комп’ютерними програмами. Призначення макросів та процедур таке.

Програмний без операндний макрос **Trn** забезпечуєблокову з 16-ти байтів пересилку від джерела в пам’яті, де розміщені ланцюги операндів в приймач, адресу яку в регістрі **DI** подає основна програма з кроком зміщення 16-ть байтів.

Програмні процедури  **Whi, Rpi** реалізують перед умовний та після умовний цикл відповідно. Процедури **Whi, Rpi** , у свою чергу, викликаються в певних місцях алгоритму макросом обслуговування ланцюгів із байтів **Srv.** Макрос **Srv** є циклічною програмою по обробці ланцюгів знакових байтів довжиною вісім. Макрос виведення повідомлення за програмним перериванням є простою програмою, тому його схема алгоритму не приводиться.

**L**

**BX:= [BX]+1**

**0**

**1**

**Proc\_Rpi**

**Al:=[BX]+len**

**Ret**

**[BX]– (AL)**

**GE**

**[BX]+len–(AL)**

**0**

**1**

**Proc\_Whi**

**Al:=[BX]**

**[BX]+len -1**

**Ret**

**Proc\_Fnc**

**Al:=[BX]+len;**

**[BX]:=[BX]⊕ al**

**Ret**

**Srv Whi**

**AKR**

**Trn 2**

**Srv Fnc**

**Trn 4**

**Srv Rpi**

**Trn 6**

**Trn 8**

**Mes1;Mes2**

**Mes\_3**

**Mes\_4**

**Mes\_5**

**Mes 6**

**AX:= Sel;**

**DS:=(AX);**

**ES:=(AX)**

**END**

**Macro\_Trn**

**SI:= Offsets X;**

**CX:= len\*2**

**TM:= [SI];**

**[DI]:=(TM) ;**

**SI:= (SI)+1;**

**DI:= (DI)+1;**

**CX:= (CX)-1**

**CX=0**

**End\_m**

**0**

**1**

**ZF=1**

**BX:= (BX)+1;**

**DI:= (DI)-1**

**0**

**1**

**Macro\_Srv**

**DI:= len;**

**BX:= Offsets X**

**Whi:Fnc:Rpi**

**End\_m**

**ZF=1**

**BX:= (BX)+1;**

**DI:= (DI)-1**

**0**

**1**

**Macro\_Srv**

**DI:= len;**

**BX:= Offsets X**

**Whi:Fnc:Rpi**

**End\_m**

**ZF=1**

**BX:= (BX)+1;**

**DI:= (DI)-1**

**0**

**1**

**Macro\_Srv**

**DI:= len;**

**BX:= Offsets X**

**Whi:Fnc:Rpi**

**End\_m**

**ZF=1**

**BX:= (BX)+1;**

**DI:= (DI)-1**

**0**

**1**

**Macro\_Srv**

**DI:= len;**

**BX:= Offsets X**

**Whi:Fnc:Rpi**

**End\_m**

**ZF=1**

**BX:= (BX)+1;**

**DI:= (DI)-1**

**0**

**1**

**Macro\_Srv**

**DI:= len;**

**BX:= Offsets X**

**Whi:Fnc:Rpi**

**End\_m**

**ZF=1**

**BX:= (BX)+1;**

**DI:= (DI)-1**

**0**

**1**

**Macro\_Srv**

**DI:= len;**

**BX:= Offsets X**

**Whi:Fnc:Rpi**

**End\_m**

**Рисунок 2 – Детальна схема алгоритму реалізації програмних циклів**

Лістинг програми, яка утворена згідно з детальним алгоритмом, достатньо повно коментована, а її складові модулі позначені певним кольором. Тому стрічки операторів програми не пояснюються мною. Програма (після її модифікації кожним студентом для свого варіанту) стандартно утилітами пакету Асемблера транслюється і компілюється з опціями ключів можливостей її виконання в середовищі програмного відлагоджувача.

Усі передбачені виведені на екран повідомлення поясняють суть вихідного алгоритму, згідно з яким розроблена асемблерна програма реалізації в комп’ютері програмних циклів. Програма чотири рази переміщає блоки з 16-ти байтів в пам’яті зі зміщенням в 16-ть байтів. Перша пересилка – переміщення вихідного стану операндів, друга передача блоку – після виконання перед умовного циклу, третя – після виконання заданої функції і третя – після виконання в комп’ютері після умовного циклу.

**Лістинг 1 – Текст програми контрольної роботи**

**.model small ;Тип моделі пам’яті**

**.stack 100h ;Сегмент стеку**

**.data ;Сегмент даних**

**X db 0, 126, 64, 1, 255, 193, 64, 192;Перший операнд**

**Y db 0, 64, 126, 0, 128, 127, 248, 15;Другий операнд**

**len equ $-Y**

**Z db len\*2\*4 dup (?)**

**mes\_1 db ' Beginning of the job processing',0dh,0ah,0ah,'$'**

**mes\_2 db ' 1 Initial state',0dh,0ah,'$'**

**mes\_3 db ' 2 State after While Y<X do Y:=Y+1',0dh,0ah,'$'**

**mes\_4 db ' 3 State after execute function X:=X or Y',0dh,0ah, '$'**

**mes\_5 db ' 4 State after Repeat X:=X-1 Until X<=Y ',0dh,0ah,'$'**

**mes\_6 db ' Completion of the job processing',0dh,0ah, '$'**

**mes\_7 db ' X:',20h,'$'**

**mes\_8 db ' Y:',20h,'$'**

**mes\_9 db ' ',0dh,0ah, '$'**

**.code ;Сегмент програми**

**srv macro Pnt ;Сервіс виконання операторів**

**local lop**

**mov cx,len ;Лічильник станів**

**mov bx,offset x ;Адреса стрічки байтів**

**lop:call Pnt ;Тестування на поточному стані**

**inc bx ;Корекція адреси**

**loop lop ;Корекція лічильника циклів**

**endm**

**Whi proc near ;Процедурна реалізація передумовного циклу**

**mov al,[bx] ;Читання елемента першого операнда**

**Whl:cmp [bx]+len,al ;Порівняння та формування усіх прапорів стану**

**jg nx1 ;Критерій ознаки порівняння Y < X**

**inc byte ptr [bx]+len ;Дія обчисленняY:=Y + 1**

**jmp Whl ;Перехід на дію**

**nx1:ret ;Повернення у основну програму**

**endp Whi**

**fnc proc near ;Процедурна реалізація функції**

**mov al,[bx]+len ;Читання елемента першого операнда**

**or [bx],al ;Функція дизюнкции**

**ret ;Повернення у основну програму**

**endp fnc**

**rpi proc near ;Процедурна реалізація функції**

**mov al,[bx]+len ;Читання елемента першого операнду**

**rip:dec byte ptr [bx] ;Декремент X:=X - 1**

**cmp [bx],al ;Порівняння та формування усіх прапорів стану**

**jge rip ;Критерій ознаки порівняння X<=Y**

**ret ;Повернення у основну програму**

**endp rpi**

**trn macro p ;Транспорт блоку з len\*p байтів**

**mov cx,len ;Лічильник циклів**

**lea si, x ;Початкова адреса**

**mov di,len\*p ;Р=2,4,6,8**

**rep movsw ;Блокова пересилка**

**endm**

**mes macro msg ;Макрос програмного**

**mov ah,09h ;виведення**

**mov dx,offset msg ;повідомлення**

**int 21h ;за перериванням**

**endm**

**scr macro xx ;Макрос видачі на екран станів CTX**

**local nxt ;Опис поміток**

**xor bx,bx ;Скид адреси**

**mov cx,8 ;Лічильник станів**

**nxt: xor ah,ah ;Очистка регістру**

**mov al, xx[bx] ;Поточний стан**

**mov dh,100 ;Дільник**

**div dh ;Ділення на 100**

**mov dl,al ;Сотні**

**mov al,ah ;Остача від ділення**

**push ax ;Збереження в стеку**

**or dl,30h ;ASCII символ сотень**

**mov ah,2 ;Функція DOS**

**int 21h ;Виведення на екран сотень**

**pop ax ;Повернення із стеку**

**xor ah,ah ;Очистка регістру**

**mov dh,10**

**div dh ;Ділення на 10**

**mov dl,al ;Десятки**

**mov al,ah ;Одиниці**

**or dl,30h ;ASCII символ десятків**

**mov ah,2 ;Функція DOS**

**push ax ;Збереження в стеку**

**int 21h ; Виведенняна екран десятків**

**pop ax ;Повернення із стеку**

**mov dl,al**

**or dl,30h ;ASCII символ одиниць**

**int 21h ;Виведення на екран одиниць**

**mov dl,20h**

**int 21h ;Виведення на екран пропуску**

**inc bx ;Наступна адреса**

**loop nxt ;Перехід на повторення**

**mov dl,20h**

**int 21h ;Виведення на екран пропуску**

**mov dl,20h**

**int 21h ;Виведення на екран пропуску**

**endm**

**exe macro ; Макрос програмного виводу станів**

**mes mes\_9 ; Курсор на початок та на нову стрічку**

**mes mes\_7 ; Назва Х**

**scr x ; Виведення в DEC форматі**

**xyz x ; Виведення в HEX форматі**

**mes mes\_9 ; Курсор на початок та на нову стрічку**

**mes mes\_8 ; Назва Y**

**scr y ; Виведення в DEC форматі**

**xyz y ; Виведення в HEX форматі**

**mes mes\_9; ; Курсор на початок та на нову стрічку**

**mes mes\_9; ; Курсор на початок та на нову стрічку**

**endm**

**scr\_sym macro ;Видача ASCII Hexmal символу**

**local fri,sym**

**cmp dl,0ah ;Порівняння з десяткою**

**jb fri ;Менше десятки**

**or dl,40h ;Формування символу**

**sub dl,09h ;велікої літері**

**jmp sym ;Перехід на видачу**

**fri:or dl,30h ;Формування символу**

**sym:int 21h ;Видача символу на екран**

**endm**

**scr\_byte macro ;Видача байту ASCII Hexmal символів**

**mov dl,[di] ;Завантаження з памяті**

**push cx ;Збереження в стеку**

**mov cl,4 ;Розрядність тетради**

**shr dl,cl ; Зсув управо на чотири**

**pop cx ;Поновлення регістру**

**scr\_sym ;Видача старшого символу байту**

**mov dl,[di] ;Завантаження того ж числа з памяті**

**and dl,0fh ;Знищення старшої тетради**

**scr\_sym ;Видача молодшого символу байту**

**mov dl,20h ;Видача символу**

**int 21h ;пропуск**

**endm**

**xyz macro adr ;Видача байтів ASCII символів змінної**

**local next**

**mov cx,8 ;Ініціалізація лічильника станів**

**lea di, adr ;та адреси**

**next:scr\_byte ;Видача байту ASCII символів**

**inc di ;Збільшення адреси**

**loop next ;Зменшення лічильника та перехід**

**mov dl,'h' ;Видача**

**int 21h ;символу h**

**endm**

**sta:mov ax,@data ;Ініціалізація**

**mov ds,ax ;регістрів на**

**mov es,ax ;перекриття сегментів**

**cld ;Обробка стрічок уперед**

**mes mes\_9 ;Для позиціювання курсору**

**mes mes\_1 ;Повідомлення про початок тестування ознаки**

**mes mes\_2 ; Повідомлення**

**exe ;Друк двох стрічок вихідного стану**

**trn 2 ;Транспорт вихідного стану**

**srv Whi ; Обчислення перед умовного циклу**

**mes mes\_3 ; Друк оператора перед умовного циклу**

**exe ;Друк двох стрічок обчислень перед умовного циклу**

**trn 4 ;Транспорт післяобчислень перед умовного циклу**

**srv fnc ; Обчислення функції**

**mes mes\_4 ; Друк оператора функції**

**exe ;Друк двох стрічок обчисленьфункції**

**trn 6 ;Транспорт після виконаної функції**

**srv rpi ; Обчислення після умовного циклу**

**mes mes\_5 ; Друк оператора після умовного циклу**

**exe ;Друк двох стрічок обчислень після умовного циклу**

**trn 8 ;Транспорт післяобчислень після умовного циклу**

**mes mes\_6 ;Повідомлення про закінчення тестування ознаки**

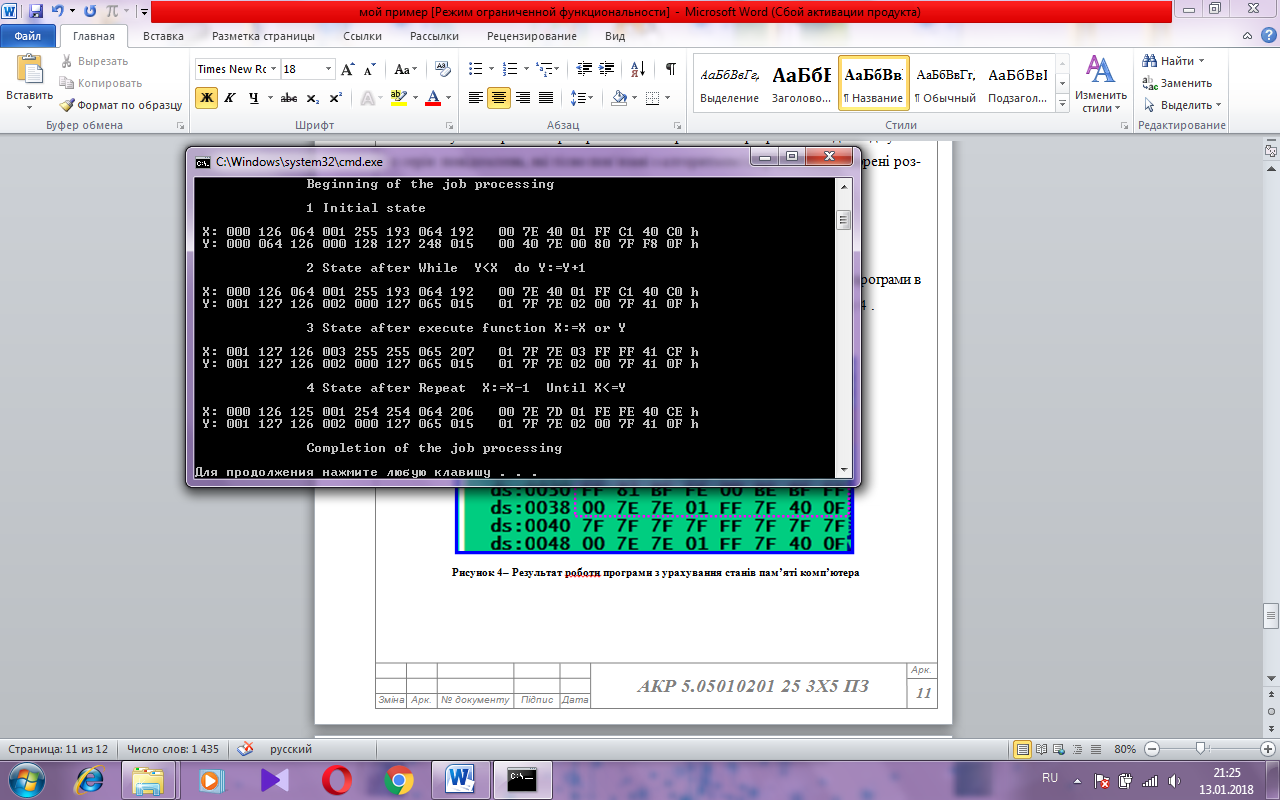
**mes mes\_9 ;Курсор на нове місце**

**mov ax,4c00h**

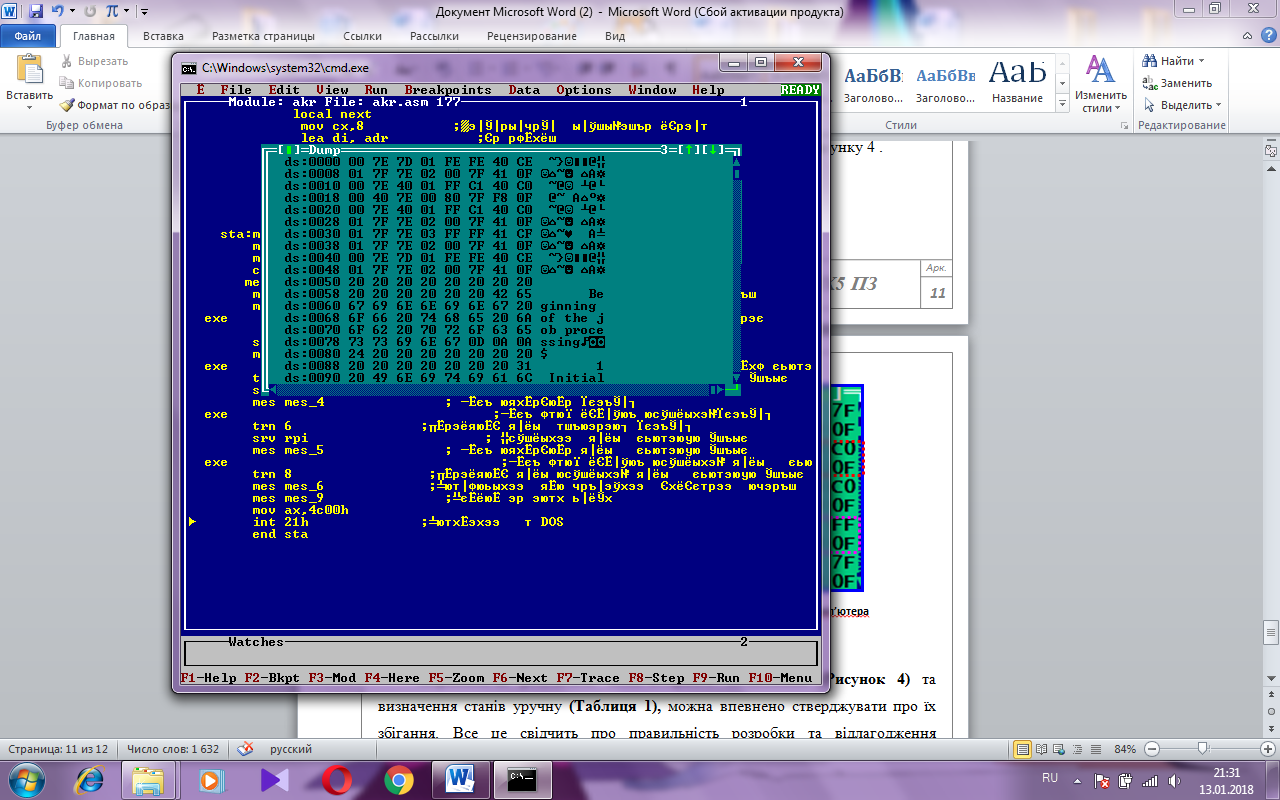
**int 21h ;Повернення в DOS**

**end sta**

Результат роботи прогарами без використання програмного відлагоджувача є серія повідомлень, які тісно пов’язані з алгоритмом і передбачено утворені розробником.

****

Результат (дамп пам’яті даних) роботи прикладної комп’ютерної програми в середовищі програмного відлагоджувача приведено унизу на Рисунку 4 .



**Рисунок 4**

Порівнюючи результати комп’ютерного обчислення **(Рисунок 4)** та визначення станів уручну **(Таблиця 1),** можна впевнено стверджувати про їх збігання. Все це свідчить про правильність розробки та відлагодження програмного забезпечення для рішення поставленої задачі. Мета виконання контрольної роботи досягнута.