2 ПРОГРАМНЕ ТЕСТУВАННЯ ПАМ'ЯТІ МІКРОПРОЦЕСОРНОЇ СИСТЕМИ

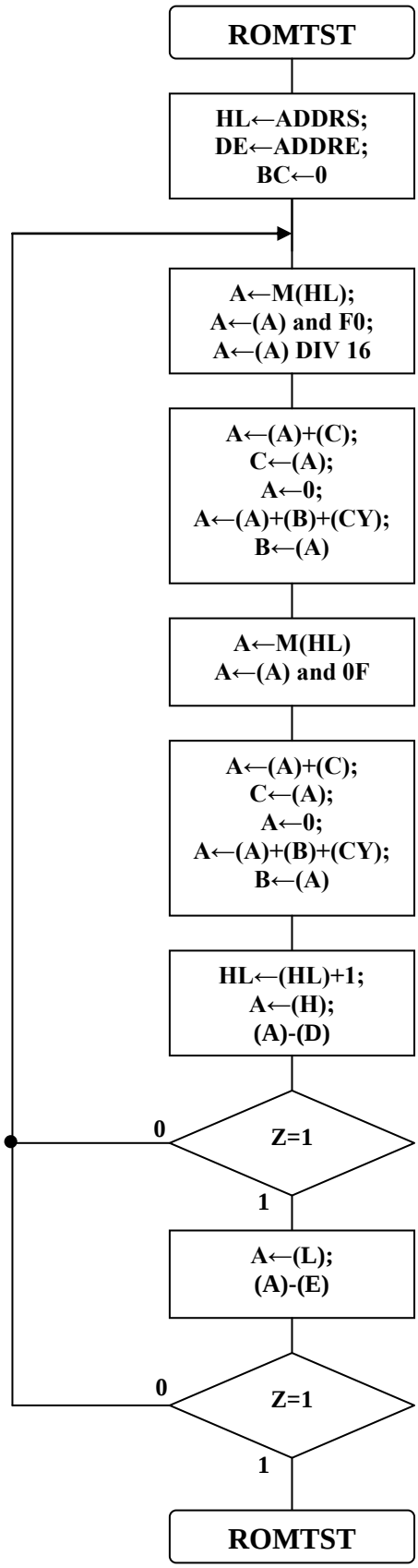
В мікропроцесорній чи комп’ютерній системі тестування виконується після запуску її в роботу. Спрацьовує самотестуюча програма, яка забезпечує перевірку роботи усіх комірок фізичної пам’яті з метою передбачено визначити вірогідність помилки в роботі усієї пам’яті. Важливу функцію виконує стек, де зберігаються початкова адреса тестуємого масиву пам’яті та кількість комірок. Багато кратне звернення до кожної комірки для запису та потім читання різних кодів допомагає визначити місце збою.

Структура тестового програмного забезпечення мало чим відрізняється від прикладного програмного продукту. Використовуються усі правила і методики в будові алгоритмів та відлагодженні програм. Програми тестування мають такі команди, як і основна програма.

2.1 Тестування програмної пам'яті мікропроцесорної системи

При відключенні енергії оперативний запам’ятовуючий пристрій свою пам'ять втрачає. Постійний запам’ятовуючий пристрій запам'ятовує практично назавжди та не залежить від джерела живлення. Постійні запам’ятовуючі пристрої особливо зручні для завдань, які потребують неодноразового повторення одного і того ж набору команд. Вони працюють зазвичай повільніше ніж оперативні, та зате їх пам'ять постійна і поміхо стійка. Крім того, свій програш в швидкості реакції постійні запам’ятовуючі пристрої компенсують щільністю упаковки.

Програмна пам'ять є одно напрямленою - читання виконується процесором, а запис в комірки неможлива. Інтерфейс модуля програмної памяті з модулем центрального процесора дуже схожий на схему підключення оперативної пам'яті до головного процесора. Винятком є не використання командного сигналу запису.



Малюнок 28 -Структурна схема алгоритму тестування програмної пам'яті

Помилка в роботі програмної пам'яті приводить до непередбачених виключень в роботі усієї мікропроцесорної чи складної комп'ютерної системи.

Алгоритм тестування програмної памяті враховує можливість головного процесора своїми регістровими парами адресування в фізичному адресному просторі комірок памяті. У **HL** і **DE** регістрових парах задається початкова і кінцева адреси **ROM (Read Only Memory)**. Обчислюється сума по модулю 2^{16} старших і молодших тетрад кожного байта окремо в заданому адресному просторі. Іншими словами, визначаються тільки два молодші байти суми. Контрольна сума утворюється у **BC** регістровій парі.

Обчислена контрольна сума процедурою тестування порівнюється в основній програмі з істиною. Якщо суми не збігаються, то присутня помилка в роботі **ROM**.

Лістинг 2 - Програма тестування програмної пам'яті

```
ROMTST:  LXI H, ADDRS          ; Ініціалізація
         LXI D, ADDRS
         LXI B, 0
NEXT:    MOV A, M              ; Читання поточного осередку
         ANI 0F0h              ; Виділення старшої тетради
         RRC
         RRC
         RRC
         RRC
         CALL SUM              ; Виклик процедури підсумовування
         MOV A, M              ; Читання поточного осередку
         ANI 0Fh               ; Виділення молодшої тетради
         CALL SUM              ; Виклик процедури підсумовування
         INX H                 ; Корекція покажчика адреси
         MOV A, H
         CMP D                 ; Порівняння з кінцевою адресою
         JNZ NEXT
         MOV A, L
```

```
        CMP E           ; Порівняння з кінцевою адресою
        JNZ NEXT
        RET
SUM:    ADD C           ; Процедура підсумовування тетрад
        MOV C, A
        MVI A, 0
        ADC B           ; Облік міжбайтового перенесення
        MOV B, A
        RET
```

Порівняння обчисленої контрольної суми в регістровій парі з еталонним значенням виконується в основній програмі шляхом віднімання. Віднімання повинно виконуватися по байтно, Спочатку обчислюються молодші байти, а потім віднімаються старші байти з урахуванням попередньої позики.

Простота та швидкодія запропонованого алгоритму є позитивною ознакою. Недоліком програми тестування постійної програмної пам’яті є відсутність ідентифікації місця помилки в робаті.

2.2 Емуляція тестування в комп’ютері оперативної пам'яті

Робота мікропроцесорної системи залежить від стану її фізичної пам’яті, де знаходяться програмний сегмент, сегменти стеку і даних. Комірки пам’яті мають розрядність байт. Тригери, які складають комірку, розміщені так близько, що на великих частотах їх перемикання спостерігається взаємодія на сосідні розряди. Тестування дозволяє передбачення такого не бажаного явища.

Тестуючі програми пам'яті мікропроцесорних систем найчастіше пишуться на мові програмування низького рівня. Асемблерні тестові програми за часом виконань найбільш швидкодіючі, що особливо важливе при самотестуванні системи після включення живлення. Для проектування програмного забезпечення тестування скори

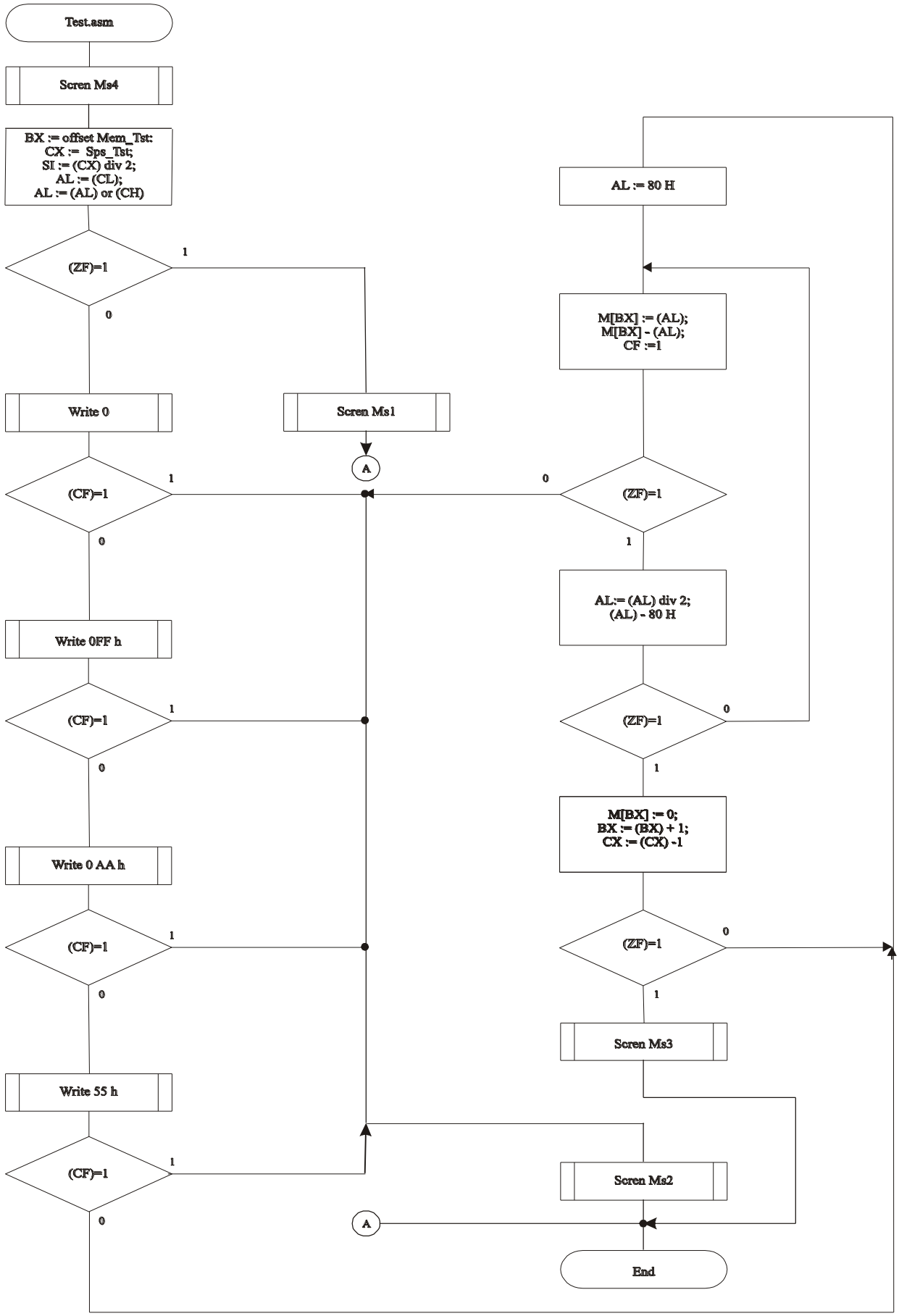
стаємося можливостями мови програмування, а саме – використання процедур і макросів. Основна програма повинна містити такі стандартні підпрограми, багатократний виклик яких дозволяє проводити тестування за короткий проміжок часу.

Процедура FilCmp заповнення тестованої області пам'яті тестовим кодом і порівняння ліченого коду з тестовим при бажанні може викликати макрос Eml_Err, що програмує помилку. Макрос запису тестового коду Write Cod здійснює виклик раніше згаданої процедури. Для супроводу вихідних даних результату тестування необхідними повідомленнями розроблений макрос Scren Msi.

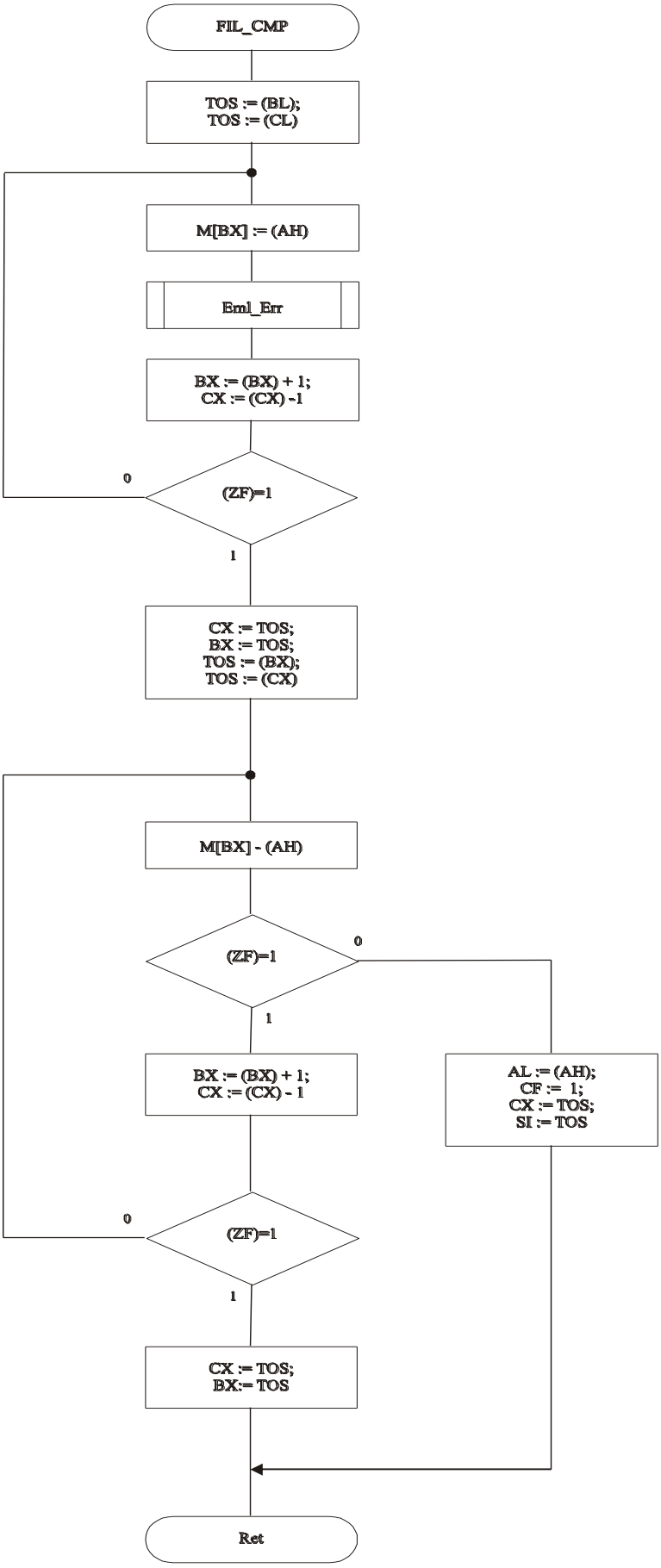
Після тестування на екран користувача видаються повідомлення про наявність або відсутність тестованої пам'яті. За наявності несправності або присутності макросу емулятора помилки указується місце несправності з подальшим перериванням тестування. Успішному виходу з тестування передує повідомлення про відсутність критичної ситуації.

Приведені на малюнках унизу детальні структурні схеми алгоритму досить повно описують процес тестування. Деталізація алгоритмів здійснена до рівня макросів і команд головного мікропроцесора, що значно спрощує процес написання асемблерної програми. У свою чергу, складена програма максимально коментована, що обертає її в добре читабельний програмний продукт.

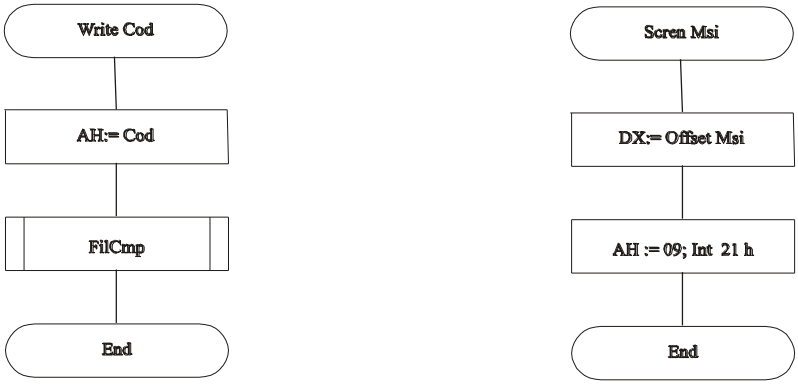
Основна програма **test.asm** чотири рази викликає макрос **write cod**, який, у свою чергу, викликає вище згадувану процедуру **filcmp**, а звідси вже може бути винесений остаточний вердикт – виклик нового макросу під ім'ям **eml_err**. У разі не емуляції помилки і працездатності всіх осередків тестованої області пам'яті здійснюється зворотне повернення в програмну крапку, звідки був здійснений виклик програми тестування.



Малюнок 29 - Алгоритми тестування оперативної пам'яті

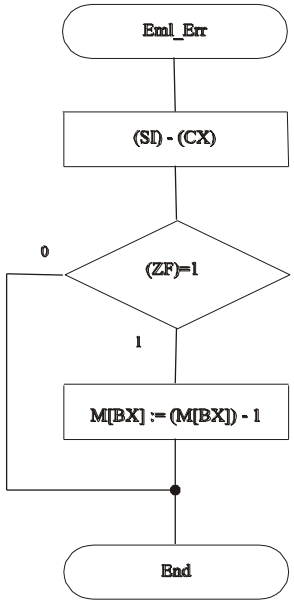


Малюнок 30 - Алгоритми процедури заповнення та порівняння



Малюнок 31 - Алгоритми макросів, що викликаються головною процедурою

Регістр АН передає в процедуру заповнення та порівняння тестуючий код, який визначається користувачем. Для текучого випадку цей код дорівнює одному із чотирьох значень.



Малюнок 32 - Алгоритми емуляції в пам'яті помилки

Помилка роботи пам'яті визначається програмними засобами і навмисно вводиться для перевірки спроектованого тестового програмного забезпечення. На стан помилки реагує основна процедура видачею на екран монітора відповідного повідомлення та блокуванням подальшого тестування.

Лістинг 3 -Програма тестування фізичної пам'яті даних

```
;test.asm
model small
.stack 100h
;Макрос обробки тестованої коди в пам'яті
write macro cod
    mov ah,cod
    call filcmp  ;Заполнение і порівняння тестової коди
    endm
;Макрос видачі повідомлень
scren macro msi
    mov ah,09h
    mov dx,offset msi
    int 21h
    endm
;Макрос емуляції помилки
eml_err macro
     local exm
     cmp si,cx
     jne exm
     dec byte ptr [bx] ;Навмисне введення помилки
  exm: endm
.data
ms1 db '              Тестуєма пам'ять відсутня',0ah,0dh,'$'
ms2 db '  Знайдена помилка за адресою DS:BX з кодом в AL',0ah,0dh,0ah,'$'
ms3 db '          Тестування проведене безпомилково',0ah,0dh,'$'
ms4 db '        Тестування   пам'яті даних мікросистеми',0ah,0dh,0ah,'$'
ms5 db '    Дипломник    XPTT                .',0ah,0dh,0ah,'$'
```

```
mem_tst db 32768 dup(?);Область пам'яті тестування
spc_tst equ 32768       ;Емність пам'яті тестування
.code
strt:jmp exe
;Процедура заповнення і порівняння тестової коди
filcmp proc
      push bx
      push cx
 fillp:mov [bx],ah ;Запись у осередок тестової коди
      ;eml_err     ;Эмуляция помилки
      inc bx      ;Коррекция адреси пам'яті
      loop fillp  ;Перехід на обробку наступного осередку
      pop cx
      pop bx
      push bx
      push cx
 cmplp:cmp [bx],ah  ;Сравнение у осередку з тестовим кодом
      jnz stop     ;Обнаружена помилка
      inc bx       ;Коррекция адреси пам'яті
      loop cmplp   ;Переход на обробку наступного осередку
      pop cx
      pop bx
      jmp yyy
  stop:mov al,ah     ;Тестовый код в AL
      stc          ;Установка помилки
      pop cx
      pop si
   yyy:ret
filcmp endp
exe:mov ax,@data         ;Инициализация сегменту даних
```

```
    mov ds,ax
    mov ax,3            ;Очистка екрану і переказ
    ;int 10h            ;курсора у його початок
    scren ms4           ;Сообщение про програму
    scren ms5           ;Сообщение про програму
    mov bx,offset mem_tst ;Начальный адреса пам'яті тестування
    mov cx,spc_tst      ;Емкость пам'яті тестування
    mov si,spc_tst/2    ;Адрес середини об'єму тестованої пам'яті
    mov al,cl           ;Проверка на факт
    or al,ch            ;наличия пам'яті тестування
    jnz next            ;Есть пам'ять тестування
    scren ms1           ;Сообщение про відсутність пам'яті тестування
    jmp xxx
next:write 0            ;Обработка тестової коди 00h
     jc errr            ;Обнаружена помилка
     write 0ffh         ;Обработка тестової коди 0ffh
     jc errr            ;Обнаружена помилка
     write 0aah         ;Обработка тестової коди 0aah
     jc errr            ;Обнаружена помилка
     write 55h          ;Обработка тестової коди 55h
     jc errr            ;Обнаружена помилка
wkl:mov al,80h          ;Поразрядно
wrl:mov [bx],al         ;в елемент пам'яті
    cmp [bx],al         ;с подальшим порівнянням
    stc                 ;Установка вірогідної помилки
    jne errr            ;Обнаружена помилка
    ror al,1            ;Модификация тестової коди, що змінюється
    cmp al,80h          ;Цикл обробки осередку
    jne wrl             ;не завершений
    mov byte ptr [bx],0 ;Обнуление поточного осередку
```

```
    inc bx              ;Переход на обробку
    loop wkl            ;очередной осередки
    scren ms3           ;Сообщение про закінчення тестування
    jmp xxx
errr:scren ms2          ;Сообщение об помилки в роботі пам'яті
xxx:mov ax,4c00h        ;Возврат у
    int 21h             ;MS DOS
    end strt
```

Процедура заповнення і порівняння тестового коду **filcmp,** що викликається, може викликати макрос емуляції помилки **eml_err**, який в елемент пам'яті, що знаходиться рівно в середині тестованого масиву, записує код помилки. Потім ця помилка програмно виявляється з подальшим текстовим повідомленням і виходом з програми тестування. Цінність цієї програми полягає не тільки в програмній емуляції помилки, що робить наочним процес тестування, але і її реальне застосування. Асемблерна програма **test.asm** після блокування виклику макросу емуляції помилки може застосовуватися для само тестування знов спроектованої мікропроцесорної системи.

Вхідними параметрами служать початкова відносна адреса тестованої області, що завантажується в регістр **BX**, і її місткість, що знаходиться в регістрі **CX** . Регістр мікропроцесора **SI** адресує середину тестованого ареалу, указуючи на осередок, що потрапив так би мовити ‘в немилість’ програмної атаки. Вихідними параметрами служать код ‘винуватого’ осередку, що передається в акумулятор **AL**, і її адресу, що повертається в регістр процесора **CX.**