Підготовка даних до програмної реалізації

Вибір режимів адресування

Для першої (початкової) реалізації виберемо такі варіанти режимів адресування:

1	регістровий	перелік допустимих регістрів для кожної команди
2	безпосередній	число може бути одно- або двобайтовим, у залежності від розміру іншого операнду
3	прямий	зміщення може бути лише *іменем*
4	неявний регістровий	[BX], [BP], [SI], [DI]
5	по базі	[BX] + зміщення - зміщення може бути лише числом без знаку; знак "+" обов'язковий
6	прямий з індексуванням	виберемо два способи запису цього адресування: TBLXX [SI] - спочатку ім'я, за ним в квадратних дужках регістр; [SI] + 8 - спочатку в квадратних дужках регістр, за ним знак "+" і число без знаку

Адресування по базі з індексуванням поки що не використовуватимемо, такий спосіб адресування можна додати на етапі розширення асемблера.

Формат проміжного файлу

Проміжний файл формується за результатами першого перегляду асемблера. В цілому проміжний файл зберігає текст асемблерної програми та всю інформацію про окремі команди, яку можна було зібрати за першим переглядом.

Кожен рядок тексту початкової програми у проміжному файлі зображається двічі:

1) копія початкового рядка без змін;

2) модифікований та розширений рядок за поданим нижче форматом, у першій позиції якого записаний знак "►" – ознака дублювання рядка. За ознаку дублювання можна вибрати й іншу літеру, яка буде відігравати роль службової і не буде зустрічатися в середині жодного рядка.

У дубльованому рядку всі маленькі латинські букви замінені на великі.

Формат дубльованого рядка:

► ; текст

►	шістнадцятково-а адреса команди / поля	шістнадцяткова довжина в байтах команди / поля	назва ко-ман-ди	фор-мат ко-манди	w – довжина регістра / пам'яті	опе-ранд 1	опе-ранд 2

Кожне окреме поле обведене прямокутною рамкою. Поля повинні розділятись пропусками. Розділення має бути або фіксованою кількістю пропусків (наприклад, двома), або довільною – не меншою одного. Зауважимо, що в дубльованому рядку непотрібне поле імені та поле коментаря.

Наприклад:

```
ACKL:  Add  bx,4     ; збільшення розміру масиву
►  001A  4  ADD  RM_RMIM  2  BX  4
```

Список повідомлень про помилки

При проектуванні програми необхідно визначити повний і точний перелік можливих помилок, які можуть бути виявлені в процесі виконання нашої програми. Для кожної можливої помилки визначаємо її цілочисловий код та текст повідомлення, який має бути виведений у файл роздруку. Наприклад, відповідно до написаних раніше алгоритмів початок такого списку може виглядати так:

101:	Закінчився текст запису асемблерної програми
102:	Літерал не закрито знаком апострофа
103:	Неправильно записано поле коментаря в команді

Таку таблицю необхідно довизначити в процесі проектування програми асемблера.

Константи для формування кодів команд

```
const
    { біти MOD визначення адресування регістра або пам'яті }  { xx000000 }
    mod0 : byte = $00 ;  { байт зміщення відсутній }
    mod1 : byte = $40 ;  { 1 байт зміщення в коді }
    mod2 : byte = $80 ;  { 2 байти зміщення в коді }
    modR : byte = $C0 ;  { біти r/m визначають регістр }

    { біти REG визначення регістра }   { 00xxx000 }
    ALr : byte = $00 ; {AL}        AXr : byte = $00 ; {AX}
    CLr : byte = $08 ; {CL}        CXr : byte = $08 ; {CX}
    DLr : byte = $10 ; {DL}        DXr : byte = $10 ; {DX}
    BLr : byte = $18 ; {BL}        BXr : byte = $18 ; {BX}
    AHr : byte = $20 ; {AH}        SPr : byte = $20 ; {SP}
    CHr : byte = $28 ; {CH}        BPr : byte = $28 ; {BP}
    DHr : byte = $30 ; {DH}        SIr : byte = $30 ; {SI}
    BHr : byte = $38 ; {BH}        DIr : byte = $38 ; {DI}

    { біт W визначення довжини регістра }
    W1 : byte = $00 ;  { однобайтовий регістр }
    W2 : byte = $01 ;  { двобайтовий регістр - слово (для 16-р. режиму) }

    { біти r/m визначення способу адресування}   { 00000xxx }
    rm_BX_SI : byte = $00 ;   { BX+SI+... }
    rm_BX_DI : byte = $01 ;  { BX+DI+... }
    rm_BP_SI : byte = $02 ;   { BP+SI+... }
    rm_BP_DI : byte = $03 ;   { BP+DI+... }
    rm_SI  :  byte = $04 ;   { SI+... }
    rm_DI  :  byte = $05 ;   { DI+... }
    rm_BP  :  byte = $06 ;   { BP+... }
    rm_BX  :  byte = $07 ;   { BX+... }

    { біти SREG визначення сегментного регістра }   { 00xxx000 }
    ESr : byte = $00 ;  { ES }
    CSr : byte = $08 ;  { CS }
    SSr : byte = $10 ;   { SS }
    DSr : byte = $18 ;   { DS }
```

Структура таблиці OPTAB

```
type
    { байти коду операції та способу адресування }
    OA = array [1..3] of byte ;
    OneFc = record    { визначення одного формату }
              NF : string[12] ;  { назва формату }
              Len : byte ; { кількість байтів коду операції та способу адресування }
              OpAdr : OA ;  { коди байтів OA }
      end ;
    { типи для різної кількості форматів }
    FC1 = record
              NameOp : string[10] ;  { назва операції }
              Lf : byte ;   { фактична кількість форматів }
              ListFc : array [1..1] of  OneFc ;  { допустимі формати }
      end ;
    FC2 = record  NameOp : string[10];  Lf : byte;  ListFc : array [1..2] of OneFc;  end ;
    FC3 = record  NameOp : string[10];  Lf : byte;  ListFc : array [1..3] of OneFc;  end ;
    FC4 = record  NameOp : string[10];  Lf : byte;  ListFc : array [1..4] of OneFc;  end ;
    FC5 = record  NameOp : string[10];  Lf : byte;  ListFc : array [1..5] of OneFc;  end ;

    { масив вказівників без типу }
    ArrPnt = array [1..{загальна кількість asm-команд}] of poiter ;
```

```
Const
    OPTAB : ArrPnt = (  @cMOV,  @cXCHG,  @cPUSH, …  ) ;
```

Визначення окремих асемблерних команд

Для визначення асемблерних команд необхідно побудувати списки або таблиці констант, що визначають окремі частини машинного коду кожної допустимої команди асемблера. Самі константи визначаються з загальної структури машинних команд (кодування регістрів, байт способу адресування, таблиці кодів команд). Загальний підхід до визначення є такий: константи повинні забезпечити швидку та ефективну побудову машинного коду з окремих відомих частин, де кожна частина отримана на основі аналізу тексту асемблерної команди. Зазвичай це досягається шляхом порозрядного (бітового) додавання значень логічною операцією OR. Наприклад, на основі введених вище означень асемблерні команди можна було б задавати так:

```
const
    cMOV :  Fc5  =
       ( NameOp : 'MOV' ;  Lf : 5 ;
         ListFc : ( ( NF : 'RM_R' ;   Len : 2 ;   OpAdr : ( $88, $00, $00 ) ),
                    ( NF : 'R_RM' ;   Len : 2 ;   OpAdr :  ( $8A, $00, $00 ) ),
                    ( NF : 'RM_IM' ;  Len : 2 ;   OpAdr :  ( $C6, $00, $00 ) ),
                    ( NF : 'SR_RM' ;   Len : 2 ;   OpAdr :  ( $8E, $00, $00 ) ),
                    ( NF : 'RM_SR' ;   Len : 2 ;   OpAdr :  ( $8C, $00, $00 ) )   ) ) ;
    cXCHG :  Fc  =
       ( NameOp : 'XCHG' ;   Lf : 1 ;
         ListFc :  ( ( NF : 'RM_R' ;   Len : 2 ;   OpAdr :  ( $86, $00, $00 ) )    ) ) ;
    cPUSH :   ……
```

Приклади для тестування асемблера

Подаємо два тестові приклади асемблерних програм, які можна використати для початкового тестування асемблера. Перший тест визначає мінімальну програму, яка друкує лише деяке стале повідомлення. Другий тест є значно більшим і визначає різні асемблерні команди, декілька способів адресування, константи та дані в різних формах, попереміннй запис команд та даних в тексті програми.

Нехай початкові тексти програм записані в файлах ProgTst1.asm та ProgTst2.asm. Наш асемблер повинен будувати такий самий машинний код, як і приведений на роздруку стандартного асемблера. Варіації можливі лише для окремих асемблерних команд щодо порядку операндів, записаних в команді, та відповідного біта *d* напрямку потоку.

Враховуючи, що команди та дані розташовані в єдиному сегменті пам'яті і об'єктний код навчального асемблера фактично відповідає COM-програмі, можна просто *назвати* отриману об'єктну програму не ProgTst1.obj, а ProgTst1.com.

Приклад 1.

```
0000                    CODESG      SEGMENT     PARA 'CODE'
                        ASSUME      CS:CODESG,DS:CODESG,SS:CODESG,ES:CODESG
                        ; мінімальні операції і виведення, нічого не читається
                        ; дані розташовані за командами
0100                                ORG   100H
0100  A1 0122 R         BEGIN:      MOV   AX,AB
0103  8B 1E 0124 R                 MOV   BX,ZN
0107  93                           XCHG  AX,BX
0108  A3 0126 R                    MOV   PRINT,AX    ; знаки +<
010B  8D 16 0126 R                 LEA   DX,PRINT    ; адреса рядка
010F  B4 09                        MOV   AH,9  ; функція DOS - вивести
                        ;                        рядок на екран
0111  CD 21                        INT   21H   ; виклик функції
0113  86 FB                        XCHG  BH,BL ; поміняти місцями букви
0115  89 1E 0126 R                 MOV   PRINT,BX    ; букви BA
0119  8D 16 0126 R                 LEA   DX,PRINT
011D  B4 09                        MOV   AH,9
011F  CD 21                        INT   21H
0121  C3                           RET         ; кінець програми
0122  4241              AB         DW    4241H ; коди літер A(41H), B(42H)
                        ;                        в оберненому порядку
0124  3C2B              ZN         DW    3C2BH ; коди знаків +(2BH),<(3CH)
                        ;                        в оберненому порядку
0126  0000              PRINT      DW    ?     ; дві літери для друкування
0128  0D                           DB    13    ; перехід на новий рядок - 13,10
0129  0A                           DB    10
012A  24                           DB    '$'   ; обмежувач рядка
                        ; мають бути надруковані на екрані два рядки:
                        ;   "+<"  і  "BA"
012B                    CODESG      ENDS
                                    END   BEGIN

    27 Source  Lines
    27 Total   Lines
    11 Symbols

 47518 + 131758 Bytes symbol space free

     0 Warning Errors
     0 Severe  Errors
```

Приклад 2.

```
0000                    CODESG      SEGMENT     PARA 'CODE'
                        ASSUME      CS:CODESG,DS:CODESG,SS:CODESG,ES:CODESG
                        ; арифметичні операції; дані поперемінно з командами
0100                            ORG   100H
0100  EB 48 90                  BEGIN:      JMP   FIRSTCM
0103  82 A2 A5 A4 F7 E2        PV1   DB    'Введіть перше число: $'
      EC 20 AF A5 E0 E8
      A5 20 E7 A8 E1 AB
      AE 3A 20 24
0119  82 A2 A5 A4 F7 E2        PV2   DB    'Введіть друге число: $'
      EC 20 A4 E0 E3 A3
      A5 20 E7 A8 E1 AB
      AE 3A 20 24
012F  82 A2 A5 A4 F7 E2        PV3   DB    'Введіть третє число: $'
      EC 20 E2 E0 A5 E2
      F5 20 E7 A8 E1 AB
      AE 3A 20 24
0145  0D                LINEFEED    DB    13    ; перехід на новий рядок
0146  0A                            DB    10
0147  24                            DB    '$'
0148  0001               COUNTNB     DW    1
014A  83 3E 0148 R 02    FIRSTCM:    CMP   COUNTNB,2   ; яке за порядком
                           число вводимо?
014F  7C 09                         JL    PRINTPV1    ; перше
0151  74 0E                         JE    PRINTPV2    ; друге
                        ; третє (JG)
0153  8D 16 012F R                  LEA   DX,PV3
0157  EB 0C 90                      JMP   FUNC
015A  8D 16 0103 R      PRINTPV1:   LEA   DX,PV1
015E  EB 05 90                      JMP   FUNC
0161  8D 16 0119 R      PRINTPV2:   LEA   DX,PV2
                        ;
0165  B4 09             FUNC:       MOV   AH,9  ; функція DOS - вивести
                        ;                         рядок на екран
0167  CD 21                         INT   21H   ; виклик функції
                        ; тепер читаємо число і переводимо у двійкову форму
0169  8D 16 01E4 R                  LEA   DX,INBUFFER
016D  B4 0A                         MOV   AH,0AH      ; функція 10 - прочитати
                        ;                               рядок до Enter
016F  CD 21                         INT   21H   ; читаємо
                        ; для простоти ціле число вводимо із знаком або без
                        ;  нього, але без пропусків спереду та в кінці
                        ;  контроль коректності запису числа не виконуємо
0171  BE 0000                       MOV   SI,0  ; індексний регістр
0174  80 3E 01E6 R 2D               CMP   BUFFX,'-'   ; чи був знак мінус
0179  74 10                         JE    MINUS
017B  80 3E 01E6 R 2B               CMP   BUFFX,'+'   ; чи був знак плюс
0180  74 13                         JE    PLUS
0182  C7 06 01F4 R 0001             MOV   ZNAK,+1     ; число без знаку - додатнє
0188  EB 15 90                      JMP   CYFRY
018B  C7 06 01F4 R FFFF MINUS:      MOV   ZNAK,-1     ; множник від'ємного
0191  46                            INC   SI   ; цифри починаються з 2-ї позиції
0192  EB 0B 90                      JMP   CYFRY
0195  C7 06 01F4 R 0001 PLUS:       MOV   ZNAK,+1     ; множник додатнього
019B  46                            INC   SI   ; цифри починаються з 2-ї позиції
019C  EB 01 90                      JMP   CYFRY
019F  A0 01E5 R         CYFRY:      MOV   AL,REALBYTES
01A2  98                            CBW
01A3  2B C6                         SUB   AX,SI ; кількість цифр
01A5  8B C8                         MOV   CX,AX ; лічильник циклів
01A7  BB 000A                       MOV   BX,10 ; множник
01AA  B8 0000                       MOV   AX,0  ; початкове значення =0
01AD  F7 E3             CONVERT:    MUL   BX    ; попереднє значення *10
01AF  50                            PUSH  AX
01B0  8A 84 01E6 R                  MOV   AL,BUFFX[SI]      ; наступна цифра
01B4  24 0F                         AND   AL,00001111B  ; виділення цифри з коду
```

```
01B6  98                          CBW
01B7  8B D0                       MOV   DX,AX
01B9  58                          POP   AX
01BA  03 C2                       ADD   AX,DX ; додали цифру
01BC  46                          INC   SI
01BD  E2 EE                       LOOP  CONVERT     ; цикл по всіх цифрах
01BF  F7 2E 01F4 R                IMUL  ZNAK  ; знак числа
                        ; запам'ятати число з регістра AX
01C3  8B 3E 0148 R                MOV   DI,COUNTNB
01C7  4F                          DEC   DI
01C8  D1 E7                       SHL   DI,1  ;  *2 - у байтах
01CA  89 85 01EE R                MOV   NUMBXX[DI],AX
                        ; всього вводимо три числа
01CE  8D 16 0145 R                LEA   DX,LINEFEED ; перейти до нового рядка
01D2  B4 09                       MOV   AH,9
01D4  CD 21                       INT   21H
                        ;
01D6  FF 06 0148 R                INC   COUNTNB
01DA  83 3E 0148 R 03             CMP   COUNTNB,3
01DF  7F 15                       JG    COMPUT
01E1  E9 014A R                   JMP   FIRSTCM
                        ; дані для читання
01E4  08                INBUFFER  DB    8     ; розмір буфера в байтах
01E5  00                REALBYTES DB    ?     ; фактично введено байтів
01E6  20 20 20 20 20 20      BUFFX DB    '         ' ; 8 пропусків - місце
                        ;                             для тексту
      20 20
01EE  0000              NUMBXX    DW    ?     ; три прочитані числа
01F0  0000                        DW    ?
01F2  0000                        DW    ?
01F4  0000              ZNAK      DW    ?
                        ;
                        ; проводимо деякі обчислення
                        ; 1) сума трьох чисел
01F6  B9 0003           COMPUT:   MOV   CX,3  ; цикл на три числа
01F9  8D 36 01EE R              LEA   SI,NUMBXX
01FD  8B 14             ADD3N:    MOV   DX,[SI]
01FF  01 16 0226 R              ADD   SUM,DX
0203  46                        INC   SI
0204  46                        INC   SI
0205  E2 F6                     LOOP  ADD3N
                        ; 2)  a*8 + b*2 + c/4    - за рахунок зсувів
0207  A1 01EE R                 MOV   AX,NUMBXX   ; a
020A  B1 03                     MOV   CL,3
020C  D3 E0                     SHL   AX,CL ; вліво на 3 розряди  ->  *8
020E  8B 1E 01F0 R              MOV   BX,NUMBXX+2 ; b
0212  D1 E3                     SHL   BX,1  ; вліво на 1 розряд  ->  *2
0214  03 C3                     ADD   AX,BX
0216  8B 1E 01F2 R              MOV   BX,NUMBXX+4 ; c
021A  B1 02                     MOV   CL,2
021C  D3 FB                     SAR   BX,CL ; вправо на 2 розряди ->  /4
021E  03 C3                     ADD   AX,BX ; остаточна сума за формулою
0220  A3 0228 R                 MOV   SUM+2,AX
0223  EB 05 90                  JMP   FORMPRINT
                        ; результати обчислень
0226  0000              SUM       DW    0     ; початкове значення суми
0228  0000                        DW    ?     ; обчислення за формулою
                        ;
                        ; друкуємо результати, перевівши числа в символьну форму
022A  8D 16 0288 R      FORMPRINT:  LEA   DX,PVDREZ
022E  B4 09                     MOV   AH,9
0230  CD 21                     INT   21H   ; заголовок результатів
0232  8B 36 02A2 R      CKLR:      MOV   SI,CNR
0236  4E                        DEC   SI
0237  D1 E6                     SHL   SI,1  ;  *2
0239  8B 84 0226 R              MOV   AX,SUM[SI]  ; AX - число для виведення
023D  3D 0000                   CMP   AX,0  ;  AX < 0 ?
0240  7D 0B                     JGE   DALI
```

```
0242  C6 06 0298 R 2D 90          MOV   OUTBUFF,'-'
0248  F7 D8                      NEG   AX    ; модуль числа
024A  EB 07 90                   JMP   MODUL
024D  C6 06 0298 R 20 90     DALI: MOV   OUTBUFF,' '
0253  BB 000A          MODUL:     MOV   BX,10 ; дільник
0256  33 F6                      XOR   SI,SI ;  SI - кількість цифр у стеку
0258  33 D2            DEFD:      XOR   DX,DX ; розширити AX до 4-байтового
025A  F7 F3                      DIV   BX
025C  80 CA 30                   OR    DL,'0'      ; DL - остача - одна цифра
025F  52                         PUSH  DX
0260  46                         INC   SI
0261  3D 0000                    CMP   AX,0  ; чи всі цифри визначені
0264  75 F2                      JNZ   DEFD
0266  8D 3E 0299 R               LEA   DI,OUTBUFF+1      ; місце на цифри
026A  8B CE                      MOV   CX,SI ; кількість цифр у стеку
026C  FC                         CLD         ; автозбільшення DI для STOSB
026D  58                LOOPC:   POP   AX
026E  AA                         STOSB       ; записати цифру в рядок
026F  E2 FC                      LOOP  LOOPC ; цикл запису цифр
0271  B0 24                      MOV   AL,'$'      ; ознака кінця рядка
0273  AA                         STOSB
0274  8D 16 0296 R               LEA   DX,PRFIX
0278  B4 09                      MOV   AH,9
027A  CD 21                      INT   21H
                      ; наступний результат
027C  FF 06 02A2 R               INC   CNR
0280  83 3E 02A2 R 02            CMP   CNR,2
0285  7E AB                      JLE   CKLR
                      ; вихід з програми
0287  C3                         RET
                      ; дані для друкування
0288  90 A5 A7 E3 AB EC      PVDREZ      DB    'Результати:  $'
      E2 A0 E2 A8 3A 20
      20 24
0296  20 20                  PRFIX       DB    '  '  ; передуючі 2 пропуски при
                      ;                                 друкуванні
0298  20 20 20 20 20 20      OUTBUFF     DB    '          '  ; 10 позицій для
                      ;                                      числа
      20 20 20 20
02A2  0001                   CNR         DW    1
                      ;
02A4                  CODESG      ENDS
                                  END   BEGIN

   158 Source  Lines
   158 Total   Lines
    39 Symbols
 47364 + 119639 Bytes symbol space free

     0 Warning Errors
     0 Severe  Errors
```