Lecture Notes on Assembly Language - J. Vaughan
Debugging a Simple Program

```
1   ; instruct.asm     An instructional version of addit.asm
2   ; assemble: nasm -f elf -l instruct.lst instruct.asm
3   ; The purpose of this file is to show what syntax errors
4   ; can arise and how they can be removed.
5   ; This file is not intended to generate executable code.
6   ; The following comments relate to the operational version
7   ; of addit.asm
8   ; addit.asm      a program for nasm for Linux, Intel, gcc
9   ;
10  ; assemble: nasm -f elf -l addit.lst addit.asm
11  ; link:     gcc -o addit addit.o
12  ; run:      addit
13  ; output:   adds 2 numbers for which num1 + num2 <= 9
14  ;          num1 and num2 are defined with equ or with db.
15  ;          sum = num1 + num2 is printed to screen
16  ;          as "The sum of <num1> and <num2> is <sum>"
17  ;
18  ; Equate Directives
19  ;
20  SPACE          equ 20h
21  LF       equ 0Ah
22  num1   equ 4
23  num1   equ 2
24  ;
25             SECTION .data
26  ; Labels to identify reg contents on screen
27  ; Decimal to ASCII translation table
28  ; Allowed hex chars here in case of error
29  decnum:        db "0123456789ABCDEF"
30  ;
31  ; Messages
32  ;
33  msg1   db "The sum of "
34  ascnum1        db SPACE
35  msg2   db " and "
36  ascnum2        db SPACE
37  msg3   db " is "
38  ascsumdb SPACE
39             db LF
40  msg1len        db $ - msg1
41  ;
42             SECTION .bss
43  sum: resb 1
44  ;
45             SECTION .text
```

```
46              global main
47      main:
48      ;
49      ; Do the arithmetic operation
50              mov al, num1
51              add al, num2
52              mov sum, al
53      ;
54      ; Translate num1 to ASCII
55              mov al, num1
56              and al, 0fh      ;Mask out upper 4 bits - should be 0 anyway
57              call asctrans
58              mov ascnum1, al        ;Ascii sum now in al
59      ;
60      ; Translate num2 to ASCII
61              mov al, num2
62              and al, 0fh      ;Mask out upper 4 bits - should be 0 anyway
63              call asctrans
64              mov ascnum2, al        ;Ascii sum now in al
65      ;
66      ; Translate sum to ASCII
67              mov al, sum
68              and al, 0fh      ;Mask out upper 4 bits - should be 0 anyway
69              call asctrans
70              mov ascsum, al         ;Ascii sum now in al
71      ;
72      ; Output message with ASCII sum
73              mov edx, msglen ; number of chars to echo on screen
74              mov ecx, msg1   ; address of string
75              mov ebx,1       ; file descriptor 1 = stdout
76              mov eax,4       ; "write" system call
77              int 0x80        ; call the kernel
78      ;
79      ; Exit to Operating System
80              mov ebx,0       ;exit with error code 0
81              mov eax,1       ;"exit" system call
82              int 0x80        ;call the kernel
83      ;
84      asctrans:
85      ; Translate a byte to 2 ASCII Chars
86      ; Source byte in AL
87      ; Destination bytes in AH,AL
88      ; Destroys: AH, AL
89      ;
90      ; Preserve ebx
91              push ebx
92      ; Zero ebx
93              xor ebx, ebx
94      ; Separate nibbles: upper into AH
95              mov ah, al
```

```
 96              shr ah, 4
 97              and al, 0xf
 98      ; Translate ah
 99              mov bl, ah
100              mov ah, [hexnum+ebx]
101      ; Translate al
102              mov bl, al
103              mov al, [hexnum+ebx]
104      ; Restore ebx
105              pop ebx
106      ;       ret
107      ;
108
```

Assembler Error Output

instruct.asm:23: error: symbol `num1' redefined

```
1       ; instruct1.asm     An instructional version of addit.asm
2       ; assemble: nasm -f elf -l instruct.lst instruct.asm
3       ; The purpose of this file is to show what syntax errors
4       ; can arise and how they can be removed.
5       ; This file is not intended to generate executable code.
6       ; The following comments relate to the operational version
7       ; of addit.asm
8       ; addit.asm       a program for nasm for Linux, Intel, gcc
9       ;
10      ; assemble: nasm -f elf -l addit.lst addit.asm
11      ; link:     gcc -o addit addit.o
12      ; run:      addit
13      ; output:   adds 2 numbers for which num1 + num2 <= 9
14      ;          num1 and num2 are defined with equ or with db.
15      ;          sum = num1 + num2 is printed to screen
16      ;          as "The sum of <num1> and <num2> is <sum>"
17      ;
18      ; Equate Directives
19      ;
20      SPACE           equ 20h
21      LF      equ 0Ah
22      num1    equ 4
23      num2    equ 2
24      ;
25              SECTION .data
26      ; Labels to identify reg contents on screen
27      ; Decimal to ASCII translation table
28      ; Allowed hex chars here in case of error
29      decnum:         db "0123456789ABCDEF"
30      ;
31      ; Messages
32      ;
33      msg1    db "The sum of "
34      ascnum1         db SPACE
35      msg2    db " and "
36      ascnum2         db SPACE
37      msg3    db " is "
38      ascsumdb SPACE
39              db LF
40      msg1len         db $ - msg1
41      ;
42              SECTION .bss
43      sum: resb 1
44      ;
45              SECTION .text
46              global main
47      main:
48      ;
49      ; Do the arithmetic operation
50              mov al, num1
```

```
51          add al, num2
52          mov sum, al
53  ;
54  ; Translate num1 to ASCII
55          mov al, num1
56          and al, 0fh     ;Mask out upper 4 bits - should be 0 anyway
57          call asctrans
58          mov ascnum1, al     ;Ascii sum now in al
59  ;
60  ; Translate num2 to ASCII
61          mov al, num2
62          and al, 0fh     ;Mask out upper 4 bits - should be 0 anyway
63          call asctrans
64          mov ascnum2, al     ;Ascii sum now in al
65  ;
66  ; Translate sum to ASCII
67          mov al, sum
68          and al, 0fh     ;Mask out upper 4 bits - should be 0 anyway
69          call asctrans
70          mov ascsum, al      ;Ascii sum now in al
71  ;
72  ; Output message with ASCII sum
73          mov edx, msglen ; number of chars to echo on screen
74          mov ecx, msg1   ; address of string
75          mov ebx,1       ; file descriptor 1 = stdout
76          mov eax,4       ; "write" system call
77          int 0x80        ; call the kernel
78  ;
79  ; Exit to Operating System
80          mov ebx,0       ;exit with error code 0
81          mov eax,1       ;"exit" system call
82          int 0x80        ;call the kernel
83  ;
84  asctrans:
85  ; Translate a byte to 2 ASCII Chars
86  ; Source byte in AL
87  ; Destination bytes in AH,AL
88  ; Destroys: AH, AL
89  ;
90  ; Preserve ebx
91          push ebx
92  ; Zero ebx
93          xor ebx, ebx
94  ; Separate nibbles: upper into AH
95          mov ah, al
96          shr ah, 4
97          and al, 0xf
98  ; Translate ah
99          mov bl, ah
100         mov ah, [hexnum+ebx]
```

```
101   ; Translate al
102             mov bl, al
103             mov al, [hexnum+ebx]
104   ; Restore ebx
105             pop ebx
106   ;         ret
107   ;
108
```

Assembler Error Output

```
instruct1.asm:52: error: invalid combination of opcode and operands
instruct1.asm:58: error: invalid combination of opcode and operands
instruct1.asm:64: error: invalid combination of opcode and operands
instruct1.asm:67: error: Unsupported non-32-bit ELF relocation
instruct1.asm:70: error: invalid combination of opcode and operands
instruct1.asm:73: error: symbol `msglen' undefined
instruct1.asm:100: error: symbol `hexnum' undefined
instruct1.asm:103: error: symbol `hexnum' undefined
instruct1.asm:111: error: phase error detected at end of assembly.
```

```
1       ; instruct2.asm     An instructional version of addit.asm
2       ; assemble: nasm -f elf -l instruct.lst instruct.asm
3       ; The purpose of this file is to show what syntax errors
4       ; can arise and how they can be removed.
5       ; This file is not intended to generate executable code.
6       ; The following comments relate to the operational version
7       ; of addit.asm
8       ; addit.asm       a program for nasm for Linux, Intel, gcc
9       ;
10      ; assemble: nasm -f elf -l addit.lst addit.asm
11      ; link:     gcc -o addit addit.o
12      ; run:      addit
13      ; output:   adds 2 numbers for which num1 + num2 <= 9
14      ;          num1 and num2 are defined with equ or with db.
15      ;          sum = num1 + num2 is printed to screen
16      ;          as "The sum of <num1> and <num2> is <sum>"
17      ;
18      ; Equate Directives
19      ;
20      SPACE           equ 20h
21      LF      equ 0Ah
22      num1    equ 4
23      num2    equ 2
24      ;
25              SECTION .data
26      ; Labels to identify reg contents on screen
27      ; Decimal to ASCII translation table
28      ; Allowed hex chars here in case of error
29      decnum:         db "0123456789ABCDEF"
30      ;
31      ; Messages
32      ;
33      msg1    db "The sum of "
34      ascnum1         db SPACE
35      msg2    db " and "
36      ascnum2         db SPACE
37      msg3    db " is "
38      ascsumdb SPACE
39              db LF
40      msg1len         db $ - msg1
41      ;
42              SECTION .bss
43      sum: resb 1
44      ;
45              SECTION .text
46              global main
47      main:
48      ;
49      ; Do the arithmetic operation
50              mov al, num1
```

```
51              add al, num2
52              mov [sum], al
53      ;
54      ; Translate num1 to ASCII
55              mov al, num1
56              and al, 0fh      ;Mask out upper 4 bits - should be 0 anyway
57              call asctrans
58              mov ascnum1, al        ;Ascii sum now in al
59      ;
60      ; Translate num2 to ASCII
61              mov al, num2
62              and al, 0fh      ;Mask out upper 4 bits - should be 0 anyway
63              call asctrans
64              mov ascnum2, al        ;Ascii sum now in al
65      ;
66      ; Translate sum to ASCII
67              mov al, sum
68              and al, 0fh      ;Mask out upper 4 bits - should be 0 anyway
69              call asctrans
70              mov ascsum, al         ;Ascii sum now in al
71      ;
72      ; Output message with ASCII sum
73              mov edx, msglen ; number of chars to echo on screen
74              mov ecx, msg1   ; address of string
75              mov ebx,1       ; file descriptor 1 = stdout
76              mov eax,4       ; "write" system call
77              int 0x80        ; call the kernel
78      ;
79      ; Exit to Operating System
80              mov ebx,0       ;exit with error code 0
81              mov eax,1       ;"exit" system call
82              int 0x80        ;call the kernel
83      ;
84      asctrans:
85      ; Translate a byte to 2 ASCII Chars
86      ; Source byte in AL
87      ; Destination bytes in AH,AL
88      ; Destroys: AH, AL
89      ;
90      ; Preserve ebx
91              push ebx
92      ; Zero ebx
93              xor ebx, ebx
94      ; Separate nibbles: upper into AH
95              mov ah, al
96              shr ah, 4
97              and al, 0xf
98      ; Translate ah
99              mov bl, ah
100             mov ah, [hexnum+ebx]
```

```
101   ; Translate al
102             mov bl, al
103             mov al, [hexnum+ebx]
104   ; Restore ebx
105             pop ebx
106   ;         ret
107   ;
108
```

Assembler Error Output

```
instruct2.asm:58: error: invalid combination of opcode and operands
instruct2.asm:64: error: invalid combination of opcode and operands
instruct2.asm:67: error: Unsupported non-32-bit ELF relocation
instruct2.asm:70: error: invalid combination of opcode and operands
instruct2.asm:73: error: symbol `msglen' undefined
instruct2.asm:100: error: symbol `hexnum' undefined
instruct2.asm:103: error: symbol `hexnum' undefined
instruct2.asm:111: error: phase error detected at end of assembly.
```

```
    1       ; instruct3.asm     An instructional version of addit.asm
 2  ; assemble: nasm -f elf -l instruct.lst instruct.asm
 3  ; The purpose of this file is to show what syntax errors
 4  ; can arise and how they can be removed.
 5  ; This file is not intended to generate executable code.
 6  ; The following comments relate to the operational version
 7  ; of addit.asm
 8  ; addit.asm      a program for nasm for Linux, Intel, gcc
 9  ;
10  ; assemble: nasm -f elf -l addit.lst addit.asm
11  ; link:     gcc -o addit addit.o
12  ; run:      addit
13  ; output:   adds 2 numbers for which num1 + num2 <= 9
14  ;          num1 and num2 are defined with equ or with db.
15  ;          sum = num1 + num2 is printed to screen
16  ;          as "The sum of <num1> and <num2> is <sum>"
17  ;
18  ; Equate Directives
19  ;
20  SPACE          equ 20h
21  LF      equ 0Ah
22  num1    equ 4
23  num2    equ 2
24  ;
25          SECTION .data
26  ; Labels to identify reg contents on screen
27  ; Decimal to ASCII translation table
28  ; Allowed hex chars here in case of error
29  decnum:        db "0123456789ABCDEF"
30  ;
31  ; Messages
32  ;
33  msg1    db "The sum of "
34  ascnum1        db SPACE
35  msg2    db " and "
36  ascnum2        db SPACE
37  msg3    db " is "
38  ascsumdb SPACE
39          db LF
40  msg1len        db $ - msg1
41  ;
42          SECTION .bss
43  sum: resb 1
44  ;
45          SECTION .text
46          global main
47  main:
48  ;
49  ; Do the arithmetic operation
```

```
50          mov al, num1
51          add al, num2
52          mov [sum], al
53  ;
54  ; Translate num1 to ASCII
55          mov al, num1
56          and al, 0fh     ;Mask out upper 4 bits - should be 0 anyway
57          call asctrans
58          mov byte[ascnum1], al       ;Ascii sum now in al
59  ;
60  ; Translate num2 to ASCII
61          mov al, num2
62          and al, 0fh     ;Mask out upper 4 bits - should be 0 anyway
63          call asctrans
64          mov byte[ascnum2], al       ;Ascii sum now in al
65  ;
66  ; Translate sum to ASCII
67          mov al, byte[sum]
68          and al, 0fh     ;Mask out upper 4 bits - should be 0 anyway
69          call asctrans
70          mov byte[ascsum], al  ;Ascii sum now in al
71  ;
72  ; Output message with ASCII sum
73          mov edx, msglen ; number of chars to echo on screen
74          mov ecx, msg1   ; address of string
75          mov ebx,1       ; file descriptor 1 = stdout
76          mov eax,4       ; "write" system call
77          int 0x80        ; call the kernel
78  ;
79  ; Exit to Operating System
80          mov ebx,0       ;exit with error code 0
81          mov eax,1       ;"exit" system call
82          int 0x80        ;call the kernel
83  ;
84  asctrans:
85  ; Translate a byte to 2 ASCII Chars
86  ; Source byte in AL
87  ; Destination bytes in AH,AL
88  ; Destroys: AH, AL
89  ;
90  ; Preserve ebx
91          push ebx
92  ; Zero ebx
93          xor ebx, ebx
94  ; Separate nibbles: upper into AH
95          mov ah, al
96          shr ah, 4
97          and al, 0xf
98  ; Translate ah
99          mov bl, ah
```

```
100             mov ah, [hexnum+ebx]
101     ; Translate al
102             mov bl, al
103             mov al, [hexnum+ebx]
104     ; Restore ebx
105             pop ebx
106     ;       ret
107     ;
108
```

Assembler Error Output

```
instruct3.asm:73: error: symbol `msglen' undefined
instruct3.asm:100: error: symbol `hexnum' undefined
instruct3.asm:103: error: symbol `hexnum' undefined
instruct3.asm:111: error: phase error detected at end of assembly.
```

```
1       ; instruct4.asm     An instructional version of addit.asm
2       ; assemble: nasm -f elf -l instruct.lst instruct.asm
3       ; The purpose of this file is to show what syntax errors
4       ; can arise and how they can be removed.
5       ; This file is not intended to generate executable code.
6       ; The following comments relate to the operational version
7       ; of addit.asm
8       ; addit.asm       a program for nasm for Linux, Intel, gcc
9       ;
10      ; assemble: nasm -f elf -l addit.lst addit.asm
11      ; link:     gcc -o addit addit.o
12      ; run:      addit
13      ; output:   adds 2 numbers for which num1 + num2 <= 9
14      ;          num1 and num2 are defined with equ or with db.
15      ;          sum = num1 + num2 is printed to screen
16      ;          as "The sum of <num1> and <num2> is <sum>"
17      ;
18      ; Equate Directives
19      ;
20      SPACE           equ 20h
21      LF      equ 0Ah
22      num1    equ 4
23      num2    equ 2
24      ;
25              SECTION .data
26      ; Labels to identify reg contents on screen
27      ; Decimal to ASCII translation table
28      ; Allowed hex chars here in case of error
29      decnum:         db "0123456789"
30      ;
31      ; Messages
32      ;
33      msg1    db "The sum of "
34      ascnum1         db SPACE
35      msg2    db " and "
36      ascnum2         db SPACE
37      msg3    db " is "
38      ascsumdb SPACE
39              db LF
40      msg1len         db $ - msg1
41      ;
42              SECTION .bss
43      sum: resb 1
44      ;
45              SECTION .text
46              global main
47      main:
48      ;
49      ; Do the arithmetic operation
50              mov al, num1
```

```
51              add al, num2
52              mov [sum], al
53      ;
54      ; Translate num1 to ASCII
55              mov al, num1
56              and al, 0fh      ;Mask out upper 4 bits - should be 0 anyway
57              call asctrans
58              mov byte[ascnum1], al          ;Ascii sum now in al
59      ;
60      ; Translate num2 to ASCII
61              mov al, num2
62              and al, 0fh      ;Mask out upper 4 bits - should be 0 anyway
63              call asctrans
64              mov byte[ascnum2], al          ;Ascii sum now in al
65      ;
66      ; Translate sum to ASCII
67              mov al, byte[sum]
68              and al, 0fh      ;Mask out upper 4 bits - should be 0 anyway
69              call asctrans
70              mov byte[ascsum], al  ;Ascii sum now in al
71      ;
72      ; Output message with ASCII sum
73              mov edx, msg1len ; number of chars to echo on screen
74              mov ecx, msg1    ; address of string
75              mov ebx,1        ; file descriptor 1 = stdout
76              mov eax,4        ; "write" system call
77              int 0x80         ; call the kernel
78      ;
79      ; Exit to Operating System
80              mov ebx,0        ;exit with error code 0
81              mov eax,1        ;"exit" system call
82              int 0x80         ;call the kernel
83      ;
84      asctrans:
85      ; Translate a byte to 2 ASCII Chars
86      ; Source byte in AL
87      ; Destination bytes in AH,AL
88      ; Destroys: AH, AL
89      ;
90      ; Preserve ebx
91              push ebx
92      ; Zero ebx
93              xor ebx, ebx
94      ; Separate nibbles: upper into AH
95              mov ah, al
96              shr ah, 4
97              and al, 0xf
98      ; Translate ah
99              mov bl, ah
100             mov ah, [decnum+ebx]
```

```
101   ; Translate al
102             mov bl, al
103             mov al, [decnum+ebx]
104   ; Restore ebx
105             pop ebx
106   ;         ret
107   ;
108
```