

The assembler for mic1 (masm) accepts strings of the form:

```
str:    "hi mom"
```

```
str1:   "hi moma"
```

and will generate output into the assembly file as:

```
str:      0110100101101000    ← left byte: i  right byte: h
          0110110100100000    ← left byte: m  right byte: space
          0110110101101111    ← left byte: m  right byte: o
          0000000000000000    ← left byte: \0 right byte: \0

str1:     0110100101101000    ← left byte: i  right byte: h
          0110110100100000    ← left byte: m  right byte: space
          0110110101101111    ← left byte: m  right byte: o
          0000000001100001    ← left byte: \0 right byte: a
```

This means that bytes are packed two per word with the last word either all full or half full of zeros as shown in the two different cases above. Since we discuss strings in this assignment it is important to understand how they are deployed in memory.

ASCII Table

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|----------|----------|-------------|----------|----------|----------|----------|----------|----------|
| 0 | NUL | DLE | space | 0 | @ | P | ` | p |
| 1 | SOH | DC1 XON | ! | 1 | A | Q | a | q |
| 2 | STX | DC2 | " | 2 | B | R | b | r |
| 3 | ETX | DC3 XOFF | # | 3 | C | S | c | s |
| 4 | EOT | DC4 | \$ | 4 | D | T | d | t |
| 5 | ENQ | NAK | % | 5 | E | U | e | u |
| 6 | ACK | SYN | & | 6 | F | V | f | v |
| 7 | BEL | ETB | ' | 7 | G | W | g | w |
| 8 | BS | CAN | (| 8 | H | X | h | x |
| 9 | HT | EM |) | 9 | I | Y | i | y |
| A | LF | SUB | * | : | J | Z | j | z |
| B | VT | ESC | + | ; | K | [| k | { |
| C | FF | FS | , | < | L | \ | l | |
| D | CR | GS | - | = | M |] | m | } |
| E | SO | RS | . | > | N | ^ | n | ~ |
| F | SI | US | / | ? | O | _ | o | del |

Ascii Table

| DEC | HEX | CH | DEC | HEX | CH | DEC | HEX | CH | DEC | HEX | CH |
|-----|------|----|-----|------|------|-----|------|----|-----|------|----|
| 000 | 0000 | ^@ | 032 | 0x20 | <sp> | 064 | 0x40 | @ | 096 | 0x60 | ` |
| 001 | 0x01 | ^A | 033 | 0x21 | ! | 065 | 0x41 | A | 097 | 0x61 | a |
| 002 | 0x02 | ^B | 034 | 0x22 | " | 066 | 0x42 | B | 098 | 0x62 | b |
| 003 | 0x03 | ^C | 035 | 0x23 | # | 067 | 0x43 | C | 099 | 0x63 | c |
| 004 | 0x04 | ^D | 036 | 0x24 | \$ | 068 | 0x44 | D | 100 | 0x64 | d |
| 005 | 0x05 | ^E | 037 | 0x25 | % | 069 | 0x45 | E | 101 | 0x65 | e |
| 006 | 0x06 | ^F | 038 | 0x26 | & | 070 | 0x46 | F | 102 | 0x66 | f |
| 007 | 0x07 | ^G | 039 | 0x27 | ' | 071 | 0x47 | G | 103 | 0x67 | g |
| 008 | 0x08 | ^H | 040 | 0x28 | (| 072 | 0x48 | H | 104 | 0x68 | h |
| 009 | 0x09 | ^I | 041 | 0x29 |) | 073 | 0x49 | I | 105 | 0x69 | i |
| 010 | 0x0a | ^J | 042 | 0x2a | * | 074 | 0x4a | J | 106 | 0x6a | j |
| 011 | 0x0b | ^K | 043 | 0x2b | + | 075 | 0x4b | K | 107 | 0x6b | k |
| 012 | 0x0c | ^L | 044 | 0x2c | , | 076 | 0x4c | L | 108 | 0x6c | l |
| 013 | 0x0d | ^M | 045 | 0x2d | - | 077 | 0x4d | M | 109 | 0x6d | m |
| 014 | 0x0e | ^N | 046 | 0x2e | . | 078 | 0x4e | N | 110 | 0x6e | n |
| 015 | 0x0f | ^O | 047 | 0x2f | / | 079 | 0x4f | O | 111 | 0x6f | o |
| 016 | 0x10 | ^P | 048 | 0x30 | 0 | 080 | 0x50 | P | 112 | 0x70 | p |
| 017 | 0x11 | ^Q | 049 | 0x31 | 1 | 081 | 0x51 | Q | 113 | 0x71 | q |
| 018 | 0x12 | ^R | 050 | 0x32 | 2 | 082 | 0x52 | R | 114 | 0x72 | r |
| 019 | 0x13 | ^S | 051 | 0x33 | 3 | 083 | 0x53 | S | 115 | 0x73 | s |
| 020 | 0x14 | ^T | 052 | 0x34 | 4 | 084 | 0x54 | T | 116 | 0x74 | t |
| 021 | 0x15 | ^U | 053 | 0x35 | 5 | 085 | 0x55 | U | 117 | 0x75 | u |
| 022 | 0x16 | ^V | 054 | 0x36 | 6 | 086 | 0x56 | V | 118 | 0x76 | v |
| 023 | 0x17 | ^W | 055 | 0x37 | 7 | 087 | 0x57 | W | 119 | 0x77 | w |
| 024 | 0x18 | ^X | 056 | 0x38 | 8 | 088 | 0x58 | X | 120 | 0x78 | x |
| 025 | 0x19 | ^Y | 057 | 0x39 | 9 | 089 | 0x59 | Y | 121 | 0x79 | y |
| 026 | 0x1a | ^Z | 058 | 0x3a | : | 090 | 0x5a | Z | 122 | 0x7a | z |
| 027 | 0x1b | ^[| 059 | 0x3b | ; | 091 | 0x5b | [| 123 | 0x7b | { |
| 028 | 0x1c | ^\ | 060 | 0x3c | < | 092 | 0x5c | \ | 124 | 0x7c | |
| 029 | 0x1d | ^] | 061 | 0x3d | = | 093 | 0x5d |] | 125 | 0x7d | } |
| 030 | 0x1e | ^^ | 062 | 0x3e | > | 094 | 0x5e | ^ | 126 | 0x7e | ~ |
| 031 | 0x1f | ^_ | 063 | 0x3f | ? | 095 | 0x5f | _ | 127 | 0x7f | ^? |

← 16 BITS →

CSR + 0

RCVR - CHARACTER IN LOWER 8 BITS

4092

+ 1

| | | | | |
|--|---|---|---|---|
| | O | I | D | B |
|--|---|---|---|---|

4093

+ 2

XMTR - CHARACTER IN LOWER 8 BITS

4094

+ 3

| | | | | |
|--|---|---|---|---|
| | O | I | D | B |
|--|---|---|---|---|

4095

The serial line controller for our machine simulation uses the layout shown above. The receiver will accept characters from a terminal device only if it has been turned ON by setting the O bit in the RCVR status register. Turning the receiver on will set B and clear D. When a character has arrived in the RCVR, D will be set and B cleared. If the I bit is set in the RCVR status register then an interrupt will be posted each time the done bit is set. The removal of a character from the RCVR register will clear the interrupt and the done bit and will set busy. If the O bit is cleared the device is effectively turned off, clearing all other bits.

The transmitter must have its O bit set to become active. When O is set the transmitter will immediately set done and force an interrupt if I is set. Any characters moved into the XMTR character register at this point will cause D to clear and will clear any outstanding interrupt. Such a character will then be drawn on the terminal display and upon completion, the D bit will be set, the B bit cleared and an interrupt will be posted if I is set. Moving another character to the transmitter will clear D and any interrupt as discussed above.

```

start: lodd on:
      stod 4095
      call xbsywt:
      loco str1:
nextw: pshi
      addd c1:
      stod pstr1:
      pop
      jzer crnl:
      stod 4094
      push
      subd c255:
      jneg crnl:
      call sb:
      insp 1
      push
      call xbsywt:
      pop
      stod 4094
      call xbsywt:
      lodd pstr1:
      jump nextw:

```

```

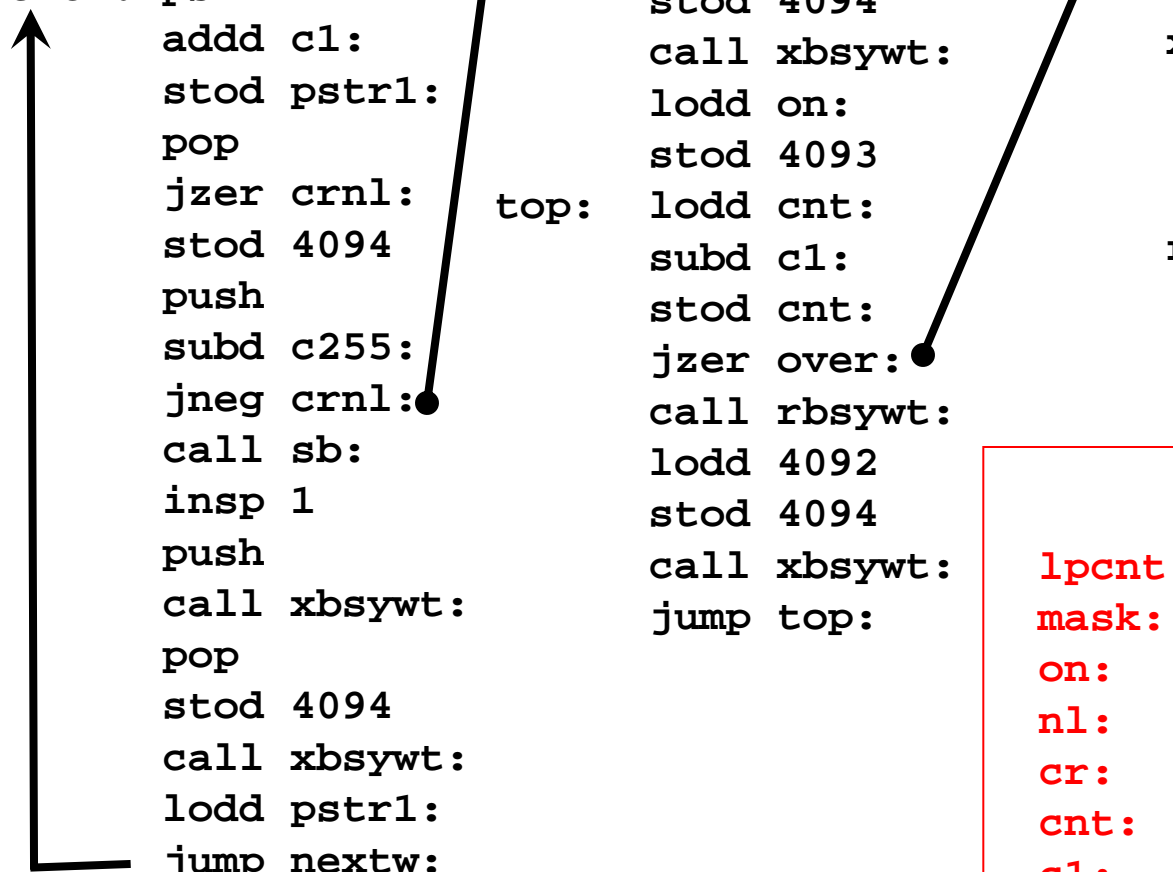
crnl: lodd cr:
      stod 4094
      call xbsywt:
      lodd nl:
      stod 4094
      call xbsywt:
      lodd on:
      stod 4093
top:   lodd cnt:
      subd c1:
      stod cnt:
      jzer over:
      call rbsywt:
      lodd 4092
      stod 4094
      call xbsywt:
      jump top:

```

```

over: lodd cr:
      stod 4094
      call xbsywt:
      lodd nl:
      stod 4094
      halt
xbsywt: lodd 4095
      subd mask:
      jneg xbsywt:
      retn
rbsywt: lodd 4093
      subd mask:
      jneg rbsywt:
      retn

```



| DATA LOCATIONS | |
|----------------|--------------------------|
| lpcnt: | 0 |
| mask: | 10 |
| on: | 8 |
| nl: | 10 |
| cr: | 13 |
| cnt: | 30 |
| c1: | 1 |
| c255: | 255 |
| pstr1: | 0 |
| str1: | "THIS IS A TEST STRING1" |

```

sb:      loco 8
loop1:   jzer finish:
         subd c1:
         stod lpcnt:
         lodl 1
         jneg add1:
         addl 1
         stol 1
         lodd lpcnt:
         jump loop1:
add1:    addl 1
         addd c1:
         stol 1
         lodd lpcnt:
         jump loop1:
finish:  lodl 1
         retn

```

DATA LOCATIONS

| | |
|--------|--------------------------|
| lpcnt: | 0 |
| mask: | 10 |
| on: | 8 |
| nl: | 10 |
| cr: | 13 |
| cnt: | 30 |
| c1: | 1 |
| c255: | 255 |
| pstr1: | 0 |
| str1: | "THIS IS A TEST STRING1" |

Ascii Table

| DEC | HEX | CH | DEC | HEX | CH | DEC | HEX | CH | DEC | HEX | CH |
|-----|------|----|-----|------|------|-----|------|----|-----|------|----|
| 000 | 0000 | ^@ | 032 | 0x20 | <sp> | 064 | 0x40 | @ | 096 | 0x60 | ` |
| 001 | 0x01 | ^A | 033 | 0x21 | ! | 065 | 0x41 | A | 097 | 0x61 | a |
| 002 | 0x02 | ^B | 034 | 0x22 | " | 066 | 0x42 | B | 098 | 0x62 | b |
| 003 | 0x03 | ^C | 035 | 0x23 | # | 067 | 0x43 | C | 099 | 0x63 | c |
| 004 | 0x04 | ^D | 036 | 0x24 | \$ | 068 | 0x44 | D | 100 | 0x64 | d |
| 005 | 0x05 | ^E | 037 | 0x25 | % | 069 | 0x45 | E | 101 | 0x65 | e |
| 006 | 0x06 | ^F | 038 | 0x26 | & | 070 | 0x46 | F | 102 | 0x66 | f |
| 007 | 0x07 | ^G | 039 | 0x27 | ' | 071 | 0x47 | G | 103 | 0x67 | g |
| 008 | 0x08 | ^H | 040 | 0x28 | (| 072 | 0x48 | H | 104 | 0x68 | h |
| 009 | 0x09 | ^I | 041 | 0x29 |) | 073 | 0x49 | I | 105 | 0x69 | i |
| 010 | 0x0a | ^J | 042 | 0x2a | * | 074 | 0x4a | J | 106 | 0x6a | j |
| 011 | 0x0b | ^K | 043 | 0x2b | + | 075 | 0x4b | K | 107 | 0x6b | k |
| 012 | 0x0c | ^L | 044 | 0x2c | , | 076 | 0x4c | L | 108 | 0x6c | l |
| 013 | 0x0d | ^M | 045 | 0x2d | - | 077 | 0x4d | M | 109 | 0x6d | m |
| 014 | 0x0e | ^N | 046 | 0x2e | . | 078 | 0x4e | N | 110 | 0x6e | n |
| 015 | 0x0f | ^O | 047 | 0x2f | / | 079 | 0x4f | O | 111 | 0x6f | o |
| 016 | 0x10 | ^P | 048 | 0x30 | 0 | 080 | 0x50 | P | 112 | 0x70 | p |
| 017 | 0x11 | ^Q | 049 | 0x31 | 1 | 081 | 0x51 | Q | 113 | 0x71 | q |
| 018 | 0x12 | ^R | 050 | 0x32 | 2 | 082 | 0x52 | R | 114 | 0x72 | r |
| 019 | 0x13 | ^S | 051 | 0x33 | 3 | 083 | 0x53 | S | 115 | 0x73 | s |
| 020 | 0x14 | ^T | 052 | 0x34 | 4 | 084 | 0x54 | T | 116 | 0x74 | t |
| 021 | 0x15 | ^U | 053 | 0x35 | 5 | 085 | 0x55 | U | 117 | 0x75 | u |
| 022 | 0x16 | ^V | 054 | 0x36 | 6 | 086 | 0x56 | V | 118 | 0x76 | v |
| 023 | 0x17 | ^W | 055 | 0x37 | 7 | 087 | 0x57 | W | 119 | 0x77 | w |
| 024 | 0x18 | ^X | 056 | 0x38 | 8 | 088 | 0x58 | X | 120 | 0x78 | x |
| 025 | 0x19 | ^Y | 057 | 0x39 | 9 | 089 | 0x59 | Y | 121 | 0x79 | y |
| 026 | 0x1a | ^Z | 058 | 0x3a | : | 090 | 0x5a | Z | 122 | 0x7a | z |
| 027 | 0x1b | ^[| 059 | 0x3b | ; | 091 | 0x5b | [| 123 | 0x7b | { |
| 028 | 0x1c | ^\ | 060 | 0x3c | < | 092 | 0x5c | \ | 124 | 0x7c | |
| 029 | 0x1d | ^] | 061 | 0x3d | = | 093 | 0x5d |] | 125 | 0x7d | } |
| 030 | 0x1e | ^^ | 062 | 0x3e | > | 094 | 0x5e | ^ | 126 | 0x7e | ~ |
| 031 | 0x1f | ^_ | 063 | 0x3f | ? | 095 | 0x5f | _ | 127 | 0x7f | ^? |