Binary Instruction Encoding

Stack-Based Code

ASCII Source Code File	Binary Encoding (in hex)
.text	0201*
main:	**--Left blank until needed for later implementation--**
push X	010202000000**
push X	010202000000
mul	030202020000
push A	010202000002
mul	030202020000
push B	010202000004
push X	010202000000
mul	030202020000
add	040202020000
push C	010202000006
add	040202020000
pop D	020202000008
END	050000000000***
.data	0200****
X: .word 3	02020000000003*****
A: .word 16	02020000020010
B: .word 5	02020000040005
C: .word 4	02020000060004
D: .word ?	02020000080000
push	05000000000000

***Signifies the .text portion of the code**

****1st byte is the opcode. 2nd byte is size of operand. Next 2 bytes are the memory segment address of operand. Last 4 bytes are the offset of the operand from its base memory segment.**

*****Signifies the "END" keyword – end of the segment.**

******Signifies the .data portion of the code.**

*******1st byte is the opcode. Next 2 bytes is the memory segment base address. Next 2 bytes are the offset of the operand from its base memory segment. Last 2 bytes is the value of the the operand.**

Accumulator-Based Code

ASCII Source Code File	Binary Encoding (in hex)
.text	0201*
main:	**--Left blank until needed for later implementation--**
load A	010202000002**
mult X	030202000000
mult X	030202000000
sto Y	020202000008
load B	010202000004
mult X	030202000000
add Y	040202000008
add C	040202000006
sto D	02020200000a
END	050000000000***
.data	0200****
X: .word 3	02020000000003*****
A: .word 16	02020000020010
B: .word 5	02020000040005
C: .word 4	02020000060004
Y: .word ?	02020000080000
D: .word ?	020200000a0000

***Signifies the .text portion of the code**

****1st byte is the opcode. 2nd byte is size of operand. Next 2 bytes are the memory segment address of operand. Last 4 bytes are the offset of the operand from its base memory segment.**

*****Signifies the "END" keyword – end of the segment.**

******Signifies the .data portion of the code.**

*******1st byte is the opcode. Next 2 bytes is the memory segment base address. Next 2 bytes are the offset of the operand from its base memory segment. Last 2 bytes is the value of the the operand.**

Program Size Calculations

- **Stack-Based Program**

  5 (8 * 2) + 13 (6 * 8) = 704 bits = 88 bytes

- **Accumulator-Based Program**

  6 (8 * 2) + 10 (6 * 8) = 556 bits = 72 bytes