# Solution for Chapter 2

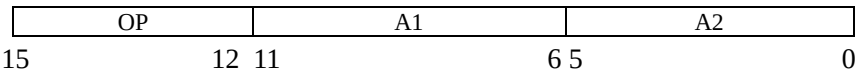
2.4 (a) 2012, (b) 5000, (c) 5028, (d) 2000, (e) 1996.

2.9 An alternative program is given below. The size of the list in bytes is computed by shifting the value *n* to the left by two bit positions, which multiplies the value by 4. This is then added to the starting address of the list to generate the address that follows the last entry in the list.
The loop in this program has only four instructions. Note that we could use a similar arrangement to process the list in the direction of increasing addresses.

| | | | |
|---|---|---|---|
| | Load | R2, N | Load the size of the list. |
| | LShiftL | R2, R2, #2 | Multiply by 4. |
| | Clear | R3 | Initialize sum to 0. |
| | Move | R4, #NUM1 | Get address of the first number. |
| | Add | R2, R2, R4 | Address past the last entry. |
| LOOP: | Subtract | R2, R2, #4 | Decrement the pointer to the list. |
| | Load | R5, (R2) | Get the next number. |
| | Add | R3, R3, R5 | Add this number to sum. |
| | Branch_if_[R4]<[R2] | LOOP | Branch back if not finished. |
| | Store | R3, SUM | Store the final sum. |

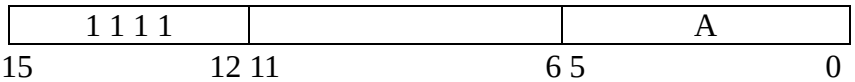
Assume that a computer's instruction length is 16-bit, and its operand address is 6-bit. Suppose the designers need two-address instructions, one-address instructions and zero-address instructions. How should we design the instruction format? And specify the numbers of each type of instruction can be designed.

(1) 二地址指令格式为：



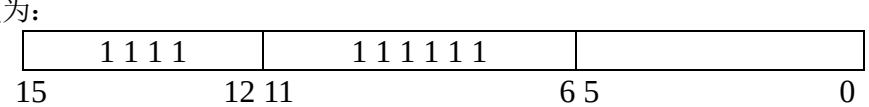

二地址指令的操作码有 4 位，将其 $2^4$=16 种编码中的 15 种 0000~1110 作为 15 条二地址指令的操作码，把剩下的编码 1111 作为非二地址指令的标志。

(2) 一地址指令格式为：




一地址指令的操作码有 10 位，高 4 位为 1111，则操作码从 1111 000000~1111 111110 共 $2^6$-1=63 种编码可表示 63 条一地址指令。剩下的编码 1111 111111 用于表示零地址指令。

(3) 零地址指令格式为：




零地址指令的操作码有 16 位，高 10 位为全 1，低 6 位从 000000~111111 共 64 种编码可表示 64 条零地址指令。

以上扩展方法可设计出二地址、一地址和零地址指令共 15+63+64=142 条。