1.2.4. IL-мова

Список інструкцій IL (Instruction List) або командний список є текстовою мовою низького рівня. Це найпростіша мова мнемонічних інструкцій, яка нагадує мову Асемблера і призначена для програмування контролерів малої потужності. Програми, що створені IL-мовою, легко транслюються в машинні коди будь-якого процесора і це дозволяє створювати дуже швидкі програми.

Проте, реально IL-мова не має ніяких переваг перед ST або FBD-мовами, оскільки проблема продуктивності вже давно розв’язана іншими методами. Тому самостійного значення не має і використовується тільки спільно з SFC-мовою. Походження – мова програмування STEP-5 німецької фірми Siemens.

1.2.4.1. Синтаксис IL-мови

IL-програма - це список послідовних інструкцій. Кожна інструкція програми починається з нового рядка і містить оператор, доповнений необов'язковими модифікаторами і, якщо необхідно для специфічних операцій, один або декілька операндів, розділених комами. Інструкції може передувати мітка з двокрапкою на кінці. Вона не є обов’язковою і ставиться тільки там, де потрібно. Якщо інструкція супроводжується коментарем, то він має бути останнім компонентом рядка. Коментар завжди починається символами '(*' і закінчується символами '*)'. Між інструкціями можуть бути введені порожні рядки, де записуються коментарі, якщо є в цьому необхідність.

Синтаксис інструкції має вигляд:

< Мітка:> < оператор> < модифікатор>< операнд> (коментар*)*

Інструкції завжди відносяться до поточного результату, який знаходиться в IL- регістрі (в асемблері це аккумулятор, а стандарт IEC 61131-3 маркірує його, як результат). Оператор визначає операцію, яка має бути виконана з поточним результатом і операндом. Результат операції знову запам'ятовується в IL- регістрі. Через те, що існує тільки один IL- регістр (поточний результат), деякі операції можуть бути затримані, таким чином, порядок виконання інструкцій може бути змінений. Для того, щоб виділити затримані операції, використовуються дужки.

В табл. 1.4 приведені IL - оператори з поясненнями і допустимими модифікаторами:

Таблиця 1.4. Оператори IL-мови

Оператор	Модифі-катор	Операнд	Опис
LD	N	Змінна, константа	Завантажує значення операнда у поточний результат
ST	N	Змінна	Запам'ятовує поточний результат у змінній. Поточний результат цією операцією не змінюється.
S		BOOL змінна	Зберігає значення TRUE в булевій змінній, якщо поточний результат

R		BOOL змінна	має значення TRUE. Ніяка операція не виконується, якщо поточний результат рівний FALSE. Поточний результат не модифікується Зберігає значення FALSE в булевій змінній, якщо поточний результат має значення TRUE. Ніяка операція не вико-нується, якщо поточний результат рівний FALSE. Поточний результат не модифікується.
AND & OR XOR	N (N (N (N (	BOOL BOOL BOOL BOOL	Логічне AND Логічне AND Логічне OR Виключне OR
ADD SUB MUL	(((	Змінна, константа Змінна, константа Змінна, константа	Складання Віднімання Множення
Продовження таблиці 1.4			
DIV	(	Змінна, константа	Ділення
GT GE EQ LE LT NE	((((((	Змінна, константа Змінна, константа Змінна, константа Змінна, константа Змінна, константа Змінна, константа	Більше: > Більше або дорівнює: >= Дорівнює: = Менше або дорівнює: <= Менше: < Не рівно: <>
CAL JMP RET	C N C N C N	Ім'я екземпляра функц. блока Мітка	Виклик функціонального блока Стрибок на мітку Закінчення виконання поточної IL програми. Якщо IL послідовність є функцією, то поточний результат повертається у визивну програму.
)			Виконує затриману операцію. Затримана операція позначається "("

Список команд має завжди починатися з оператора LD (команда завантаження IL- регістра) і закінчуватися оператором збереження ST.

Приклад:

```
LD   20
SUB 8
ST    M
```

Тут в IL- регістр (акумулятор) завантажується число 20, з якого віднімається 8, а результат запам'ятовується у змінній M. Після цього вміст змінної та IL- регістра (акумулятора) дорівнює 12.

Операції порівняння також завжди виконуються з акумулятором (IL-регістром), а булевий результат дії залишається в акумуляторі.

Приклад:

```
LD  M
LT  5
```

Тут значення змінної M завантажено у акумулятор і порівнюється з числом 5. Якщо M більше або дорівнює 5, вміст акумулятора (IL- регістра) або результат - 0 (FALS), якщо менше 5 – 1 (TRUE).

Приклади використання деяких операторів IL-мови :

Оператора LD

```
LD    false       (* результат в IL- регістрі:= булева константа FALSE *)
LD    true        (* результат в IL- регістрі := булева константа TRUE *)
LD    123         (* результат в IL- регістрі := цілочислова константа *)
LD    123.1       (* результат в IL- регістрі := дійсна константа *)
LD    t#3ms       (* результат в IL- регістрі:= часова константа *)
LD    boo_var1    (* результат в IL- регістрі := булева змінна *)
LD    ana_var1    (* результат в IL- регістрі:= цілочислова змінна *)
LD    tmr_var1    (* результат в IL- регістрі:= часова змінна *)
```

Оператора ST:

```
LD    false       (* поточний результат в IL- регістрі:= FALSE *)
ST    boo_var1 (* boo_var1 := FALSE *)
LD    123         (* поточний результат в IL- регістрі:= 123*)
ST    ana_var1 (* ana_var1 := 123 *)
LD    t#12s       (* поточний результат в IL- регістрі:= t#12s*)
ST    tmr_var1 (* tmr_var1 := t#12s *)
```

Оператора S:

```
LD   true        (* поточний результат в IL- регістрі:= TRUE *)
S    boo_var1 (* boo_var1 := TRUE *)
                 (* поточний результат в IL- регістрі не змінюється *)
LD   false       (* поточний результат в IL- регістрі:= FALSE *)
S    boo_var1 (* нічого не зроблене - boo_var1 не змінюється *)
```

Оператора R:

```
LD  true       (* поточний результат в IL- регістрі:= TRUE *)
R   boo_var1 (* boo_var1 := FALSE *)
               (* поточний результат в IL- регістрі не змінюється *)
ST  boo_var2 (* boo_var2 := TRUE *)
LD  false      (* поточний результат в IL- регістрі:= FALSE *)
R   boo_var1 (* нічого не зроблене - boo_var1 не змінюється *)
```

Додавання до мнемоніки деяких операторів символів - модифікаторів 'C', 'N' і '(' модифікує значення інструкції. Символ модифікатора завершує ім'я оператора без пропуску, наприклад: JMPC, ANDN, CALCN.

Модифікатор 'N' (negation) інвертує значення операнда до виконання інструкції. Операнд має бути типів BOOL, BYTE, WORD або DWORD.

Приклад:

```
LDN    boo_var2   (* результат := NOT ( булева змінна ) *)
```

Модифікатор 'C' (condition) вказує на те, що відповідна інструкція має бути виконана тільки за умови, що поточний результат має значення TRUE (відмінне від 0 для небулевих значень).

Приклад:

```
JMPC    test1     (*виконується тільки, коли результат:= TRUE*)
```

Модифікатор 'C' може комбінуватися з модифікатором 'N'. Тоді інструкція виконується тільки за умови, що поточний результат має значення FALSE (або 0 для не булевих значень).

Приклад:

```
CALCN   trigb1    (*виконується тільки коли результат:= FALSE*)
```

Модифікатор відкриваюча дужка '(' вказує на те, що оцінка інструкції має бути затримана до тих пір, поки не зустрінеться оператор закриваюча дужка ')', який виконує затриману операцію.

Приклад:

```
(* res := a1 + (a2 * (a3 - a4) * a5) + a6; *)
LD   a1 (* результат := a1; *)
ADD( a2 (* затримане складання ADD - результат := a2; *)
MUL( a3 (* затримане множення MUL - результат := a3; *)
SUB  a4 (* результат := a3 - a4; *)
)       (* виконання затриманого множ. MUL - результат:=a2*(a3-a4); *)
MUL  a5 (* результат := a2 * (a3 - a4) * a5; *)
)       (* виконання затриманого складання ADD *)
        (* результат := a1 + (a2 * (a3 - a4) * a5); *)
ADD  a6 (* результат := a1 + (a2 * (a3 - a4) * a5) + a6; *)
ST    res  (* збереження поточного результату в змінній res *)
```

Інструкції може передувати мітка з двокрапкою (':'). Мітка використовується при необхідності переводу виконання програми на вказану

інструкцію. Щоб перехід відбувся, необхідно використати операцію стрибка, а в якості операнда вказати ім’я мітки. При цьому:

- ім'я не може бути довшим 16 символів;
- першим символом повинна бути буква;
- подальшими символами можуть бути букви, цифри або символ підкреслення ('_').

У одній IL-програмі те саме ім'я не може бути використане для позначення більше однієї мітки. Ім'я мітки може співпадати з ім'ям змінної.

Операнд і поточний вміст акумулятора (IL-регістра) повинні мати однаковий тип даних. Якщо є потреба опрацювати операнди різних типів даних, то спочатку здійснюється перетворення типів за допомогою наступних операторів:

ANY_TO_DINT-Перетворити будь-яку змінну в цілу;
ANY_TO_REAL-Перетворити будь-яку змінну в дійсну;
ANY_TO_SINT- Перетворить будь-яку змінну в коротку цілу;
ANY_TO_STRING-Перетворити будь-яку змінну в рядок;
ANY_TO_TIME-Перетворить будь-яку змінну в часову.

Виключенням є тип даних TIME разом з арифметичними операторами MUL і DIV. Ці оператори дозволяють опрацьовувати операнд типу даних TIME разом з операндом типу даних ANY_NUM. У цьому випадку результат цих команд має тип даних TIME.

Приклад:

Перевірити значення цілочислового селектора (0, 1 або 2) і за допомогою оператора JMPC встановити булевий вихід, що дорівнює 0.

```
JMPex:  LD              selector    (* селектор - 0 або 1 або 2 *)
        ANY_TO_BOOL                 (* перетворення у Boolean *)
        JMPC            test1       (* if selector = 0 then *)
        LD              true
        ST              bo0         (* bo0 := true *)
        JMP             JMPend      (* кінець програми *)
test1:  LD              selector
        SUB             1           (* зменшити селектор: тепер 0 або1 *)
        ANY_TO_BOOL                 (* перетворення у Boolean *)
        JMPC            test2       (* if selector = 0 then *)
        LD              true
        ST              bo1         (* bo1 := true *)
        JMP             JMPend      (* кінець програми *)
test2:  LD              true        (* остання можливість *)
        ST              bo2         (* bo2 := true *)

JMPend:                             (* кінець IL програми *)
```

При IL-програмуванні можна використовувати функції та функціональні блоки.

Виклик функції в IL (написаної на будь-якій з мов IL, ST, LD, FBD або 'C') використовує її ім'я як оператора. При цьому перший параметр виклику має бути збережений у поточному результаті перед викликом. Решта параметрів записується в полі операнда, розділяючись комами. Після виконання функції, значення що нею повертається, запам'ятовується у поточному результаті IL.

Приклад:

```
(*Визивна функція : перетворити цілочислове значення в часове *)
Main:  LD          bi0
       MYFUNC      (* виклик функції для отримання цілочислового
                   значення *)
       ST          (* результат := значення, що повертається
                   функцією *)
       GT          (* перевірка переповнення значення*)
       RETC        (* повернення по переповненню *)
       LD
       MUL         (* перетворити секунди в мілісекунди *)
       ANY_TO_TIME (* перетворити в таймер *)
       ST          (* запам'ятати перетворене значення в таймері *)
```

Функція, що викликається з ім'ям MYFUNC, оцінює цілочислове значення, задане як двійкове число трьох булевих входів: in0, in1, in2 - три вхідні параметри функції:

```
LD              in2
ANY_TO_DINT                 (* result = ANY_TO_DINT (in2); *)
MUL             2           (* result := 2 * ANY_TO_DINT (in2); *)
ST              temporary   (* temporary := result *)
LD              in1
ANY_TO_DINT
ADD             temporary   (* result := 2 * ANY_TO_DINT (in2) +
                            ANY_TO_DINT (in1); *)
MUL             2           (* result := 4 * ANY_TO_DINT (in2) + 2 *
                            ANY_TO_DINT (in1); *)
ST              temporary   (* temporary := result *)
LD              in0
ANY_TO_DINT
ADD             temporary   (* result := 4 * ANY_TO_DINT (in2) + 2 *
                            ANY_TO_DINT (in1) + ANY_TO_DINT (in0); *)
```

Виклик функціонального блока здійснюється за допомогою оператора CAL, при цьому в якості операнда використовується ім'я екземпляра функціонального блока.

Вхідні параметри блока повинні бути призначені до виклику за допомогою послідовності дій LD/ST. Вихідні параметри, якщо використовуються, відомі[4,7-9].

Приклад:

Викликати функціональний блок SR : SR1 є екземпляром SR

```
LD      auto_mode
AND     start_cmd
ST      SR1.set1
LD      stop_cmd
ST      SR1.reset
CAL     SR1
LD      SR1.Q1
ST      command
```

1.2.4.2. Приклад програмування IL-мовою

Як приклад програмування IL – мовою розробимо програму керування роботою двох двигунів згідно з алгоритмом наведеним у параграфі 1.2.2.2.

Проектний код IL-мовою має вигляд:

```
(*Програмування запуску двигуна_1*)
LD  start_1
S   motor_1
(*Програмування таймера роботи двигуна_1*)
LD  motor_1
ST  M_TIME_1.IN
LD  T#5s
ST  M_TIME_1.PT
CAL M_TIME_1
LD  M_TIME_1.Q
ST  time_1
LD  time_1
R   motor_1
S   motor_2
LD  M_TIME_1.ET
ST  act_time_1
(*Програмування лічильника натисків кнопки запуску двигуна_2*)
LD  start_2
ST  MOT_COUNT.CU
LD  motor_2
ST  MOT_COUNT.RESET
LD  int#2
ST  MOT_COUNT.PV
CAL MOT_COUNT
LD  MOT_COUNT.Q
S   motor_2
LD  MOT_COUNT.CV
ST  pressed
```

```
ST  pressed
(*Програмування таймера роботи двигуна_2*)
LD  motor_2
ST  M_TIME_2.IN
LD  t#10s
ST  M_TIME_2.PT
CAL M_TIME_2
LD  M_TIME_2.Q
ST  time_2
LD  time_2
R   motor_2
S   motor_1
LD  M_TIME_2.ET
ST  act_time_2
(*Програмування зупинення двигуна_1 і двигуна_2*)
LD  stop
R   motor_1
R   motor_2
(*Програмування підрахунку циклів роботи двигунів та їх
зупинення*)
LD  time_1
OR  time_2
ST  Cycle_Count.CU
LD  Reset
ST  Cycle_Count.RESET
LD  int#10
ST  Cycle_Count.PV
CAL Cycle_Count
LD  Cycle_Count.Q
R   motor_1
R   motor_2
LD  Cycle_Count.CV
ST  Motor_Cycles
```

Тут:

START_1 і START_2 - кнопки початкового пуску двигунів;

MOTOR_1 і MOTOR_2 – перший і другий двигуни;

M_TIME_1 і M_TIME_2 – екземпляри функціонального блока таймера TON, які відлічують час роботи двигунів;

MOT_COUNT – екземпляр функціонального блока лічильника CTU, який підраховує кількість натисків на кнопку START_2;

CYCLE_COUNT(*CYCLE_COUNT:=count+1;)*– функція користувача, яка підраховує кількість активізацій двигунів;

STOP – кнопка непередбаченої зупинки двигунів.