SUPORTE DE CURSO

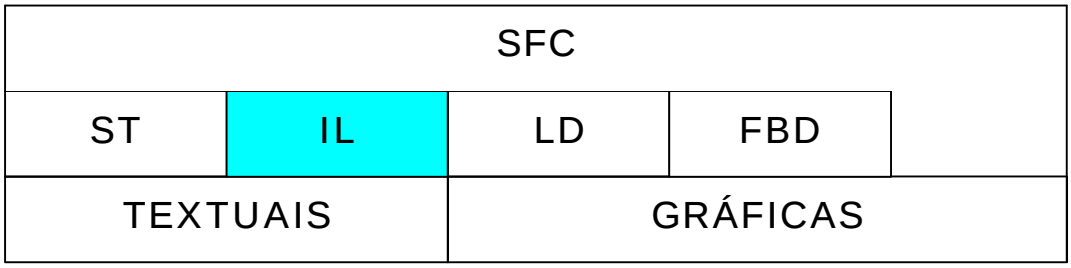
IEC 1131-3 Instruction List

Livro Texto: Programming industrial control systems using IEC 1131-3 – R.W. Lewis

UFMG – Informática Industrial
Prof. Constantino Seixas Filho

IEC 1131-3 Instruction List

Structured Text (ST)	Textuais
Instruction List (IL)	
Function Block Diagram (FBD)	Gráficas
Ladder Diagram (LD)	
Sequential Function Charts (SFC)	



É uma linguagem textual, de baixo nível, usada para descrever o comportamento de :

- Funções
- Blocos de funções
- Programas
- Em SFC para expressar o comportamento, ações e transições.

Princípios básicos

- Linguagem textual de baixo nível semelhante ao assembly. Ideal para resolver problemas simples e pequenos onde existem poucas quebras no fluxo de execução do programa.

- A linguagem é mais fácil de implementar que o texto estruturado e é a linguagem preferida por pequenos implementadores de CLPs.

- É considerada por muito implementadores como a linguagem para a qual todas as demais linguagens devem ser traduzidas. Este não é entretanto um requerimento do standard.

- Outros a encaram como a linguagem preferencial de pequenos CLPs e enxergam o Texto Estruturado como a linguagem base.

- Na verdade é apenas uma linguagem adicional, menos amigável e flexível e que deve ser usada para produzir código otimizado para trechos de performance crítica em um programa.

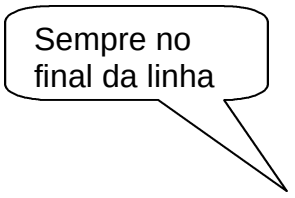
Estrutura da linguagem

Cada instrução ocupa uma linha

Instrução = Operador + Operandos

Cada instrução usa ou muda o valor de um único registrador denominado registro de resultado ou acumulador.

Exemplo:



Label	Operador	Operando	Comentário
	LD	Velocidade	(* Carregue Velocidade e *)
	GT	1000	(* Teste se é maior que 1000 *)
	JMPCN	VOLTS_OK	(* Jump se não é *)
	LD	Volts	(* Carregue Volts *)
	SUB	10	(* Retirar 10 *)
	ST	Volts	(* Armazenar em Volts *)
VOLTS_OK:	LD	1	(* Carregue 1 e armazene *)
	ST	%Q75	(* na saída 75 *)

O programa acima é equivalente ao seguinte trecho em Texto Estruturado:

```
IF Velocidade > 1000 THEN
        Volts := Volts - 10;
END_IF;
%Q75 = 1;
```

Semântica das Instruções:

Instruções Gerais:

Operador Operando eqüivale a :

Acumulador := Acumulador Operador Operando

Exemplo:

SUB 10 ≡ Acumulador := Acumulador SUB 10

A instrução Load:

LD Acumulador := Operando

Exemplo:

LD Velocidade ≡ Acumulador := Velocidade

A instrução Store:

ST Operando := Acumulador

Exemplo:

ST VOLTS ≡ VOLTS := Acumulador

Execução adiada

Algumas instruções permitem o uso do modificador parêntesis : (

Este modificador permite que resultados intermediários sejam obtidos sem alteração do acumulador. O efeito é o mesmo de se usar parêntesis em expressões aritméticas normais e em expressões booleanas.

Exemplo: execução adiada

Operador	Operando	Comentário
LD	A	(* Adicione A e B *)
ADD	B	(* Mantenha o resultado no acumulador *)
MUL(	A	(* Adie até (A-B) ser calculado *)
SUB	B	
)		(* agora multiplique por (A-B) *)

Quando o primeiro parêntesis, depois de MUL é alcançado, o resultado de A + B é mantido no acumulador principal e A é carregado num acumulador temporário. A operação de MUL será adiada até que o fecha parêntesis seja alcançado. SUB B produz A - B que é mantido no acumulador temporário. ')' é alcançado e o valor do acumulador temporário é multiplicado pelo valor do acumulador principal.

Este programa é equivalente a :

resultado := (A + B) * (A - B)

Não existe precedência operadora com os operadores IL

O exemplo abaixo realiza a operação:

X + (B * (C + D))

Operador	Operando	Comentário
LD	X	(* Carregue X *)
ADD(	B	(* Adie ADD, carregue B *)
MUL(	C	(* Adie MUL, carregue C *)
ADD	D	(* Some D *)
)		(* multiplique acumulador *)
)		(* some *)

Este conjunto de operações é equivalente às seguintes operações em uma pilha :

PUSH X
PUSH B
PUSH C
PUSH D
ADD
MUL
ADD

Operações:

PUSH X	PUSH B	PUSH C	PUSH D	ADD	MUL	ADD

Situação da Pilha:

TOS ⇨

X	B	C	D	D+C	B*(D+C)	X + B*(D+C)
	X	B	C	B	X	
		X	B	X		
			X			

Não use *jumps* dentro de uma seção entre parêntesis.

Modificadores:

N - Nega um valor booleano.
C - Denota jump condicional.
Só pode ser utilizado com a instrução JMP.

Exemplo:

```
LD      %IX10     (* Carrega Entrada 10 *)
ANDN    Switch1   (* AND NOT Switch1 *)
JMPNC   Label1    (* Jump se não é verdadeiro *)
```

Operadores

Operador	Modificador	Operando	Comentários
LD	N	Qualquer[1]	Carrega operando no acumulador
ST	N	Qualquer[1]	Armazena acumulador no operando
S		**BOOL**	Seta operando para TRUE
R		**BOOL**	Reseta operando para False
AND	N,(	**BOOL**	AND booleano
&	N,(	**BOOL**	equivalente a AND
OR	N,(	**BOOL**	OR booleano
XOR	N,(	**BOOL**	OU exclusivo
ADD	(	Qualquer[1]	Adição
SUB	(	Qualquer[1]	Subtração
MUL	(	Qualquer[1]	Multiplicação
DIV	(	Qualquer[1]	Divisão

[1] Pode representar qualquer tipo simples: SINTY, INT, DATE_AND_TIME, **REAL**, etc.

Operadores comparativos e de desvio (*jump*):

Operador	Modificador	Operando	Comentários
GT	(	Qualquer	Comparação maior que
GE	(	Qualquer	Comparação maior ou igual
EQ	(	Qualquer	Comparação igual
NE	(	Qualquer	Comparação diferente
LE	(	Qualquer	Comparação menor ou igual
LT	(	Qualquer	Comparação menor
JMP	C,N	Label	Salta para label
CAL	C,N	Nome	Chamada de bloco de função
RET	C,N		Retorno de função ou bloco de função
)			Execute o último operador adiado

Os operadores que possuem dois modificadores podem ser usados com ambos:

Operador	Semântica
AND	AND booleano
AND(	AND adiado
ANDN(	Adia a execução e inverte o resultado (NAND adiado)
ANDN	Função NAND

Chamada de funções e blocos de funções:

Existem três diferentes formatos de chamada:

Usando lista de entrada:

```
CAL LOOP1(SP:= 300.0, PV := %IW20)
```

Carregando as entradas antes da chamada

```
LD    300.0
ST    LOOP1.SP
LD    %IW20
ST    LOOP1.PV
CAL   LOOP1
```

Usando operadores de entrada

Apenas alguns blocos de funções permitem chamada direta através de instruções especiais em IL.

Exemplo:

Biestável ST (*Set/reset*) e bloco de contagem *up/down* (CTU).

```
S1    Latch1    (* Seta instância do bloco de função SR *)
LD    10        (* Acumulador = 10 *)
PV    CTU1      (* Copia 10 no parâmetro PV do contador CTU1 *)
CU    CTU1      (* Chama contador up/down habilitado para contar *)
```

Operadores de blocos de função:

Operador	Tipo de bloco de função	Comentários
S1, R	Biestável SR	Seta e Reseta o biestável SR
S, R1	Biestável RS	Seta e Reseta o biestável RS
CLK	R_Trig, detector de borda de subida	Entrada de clock do bloco lógico detector de borda de subida.
CLK	F_Trig, detector de borda de descida	Entrada de clock do bloco lógico detector de borda de descida.
CU, R, PV	CTU, contador incremental	Parâmetros de controle para o bloco de controle CTU contador incremental: CU incrementa, R Reset e PV Carrega contador.
CD, LD, PV	CTU, contador decremental	Parâmetros de controle para o bloco de controle CTD contador decremental: CD incrementa, LD Carrega e PV Carrega contagem mínima.
CU, CD, R, LD, PV	CTUD, contador universal	Parâmetros de controle para o bloco de controle CTUD contador universal.
IN, PT	TP, temporizador de pulso	Parâmetros de controle para o timer de pulso: IN inicia temporização, PT seta o tempo de pulso.
IN, PT	TON, temporizador de atraso de subida.	Parâmetros de controle para o timer de atraso de subida: IN inicia temporização, PT seta o tempo de pulso.
IN, PT	TOF, temporizador de atraso de descida.	Parâmetros de controle para o timer de atraso de descida: IN inicia temporização, PT seta o tempo de pulso.

Deficiências do padrão IL:

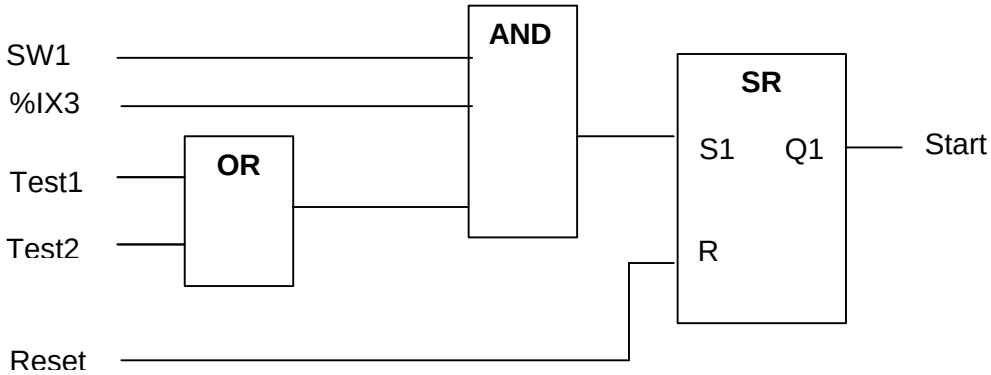
- Não descreve o comportamento da máquina virtual que executará a linguagem.
- Não explica como o acumulador armazenará dados de tipos diferentes.
- Não explica como o acumulador armazenará dados estruturados: matrizes ou vetores.

- Não descreve como erros de run-time são tratados.
- Padrão deverá ser revisto.

Portabilidade entre IL e outras linguagens IEC

Tradução de outras linguagens para IL é mais fácil que de IL para outras linguagens.

Exemplo:



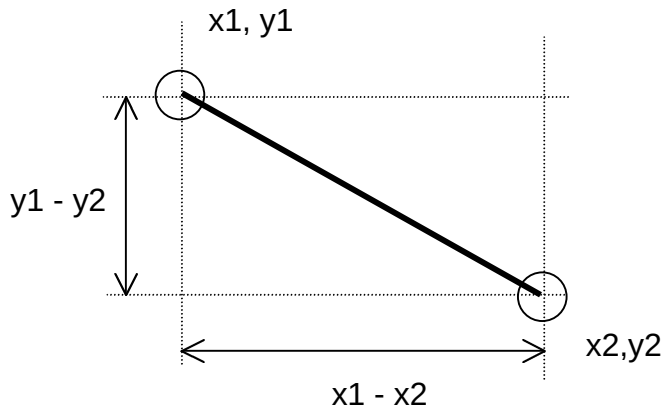
Programa em IL:

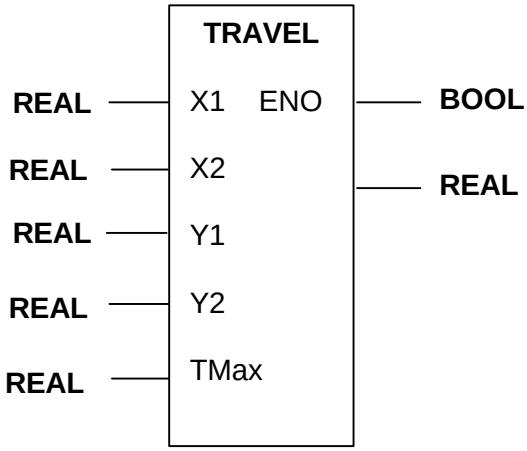
Operador	Operando	Comentário
LD	Test1	(* Acumulador = Test1 *)
OR	Test2	(* Acc = Test1 OR Test2 *)
AND	SW1	(* AND SW1 *)
AND	%IX3	(* AND input 3 *)
ST	StartSR.S1	(* Seta entrada de StartSR *)
LD	Reset	(* Carrega valor de Reset *)
ST	StartSR.R	(* Armazena na entrada Reset *)
CAL	StartSR	(* Chama bloco de função StartSR *)
LD	StartSR.Q1	(* Carrega saída Q1 *)
ST	Start	(* e armazena em Start *)

Definindo funções e blocos de funções

- IL pode ser usado para definir funções e bloco de funções.
- Quando usada para definir uma função, o valor retornado é último valor no acumulador.

Exemplo: Distância de deslocamento entre dois furos em uma superfície plana.





Distância = SQRT((X1 - X2) * (X1 - X2) + (Y1 - Y2) * (Y1 - Y2));

A saída EN0 deve indicar se o valor de distância é menor que Tmax, o alcance máximo da máquina.

```
FUNCTION TRAVEL: REAL
VAR_INPUT
        X1, X2, Y1, Y2: REAL; (* coordenadas dos pontos extremos *)
        Tmax: REAL;
END_VAR
VAR
        temp: REAL; (* Valor temporário *)
END_VAR

        LD    Y1
        SUB   Y2          (* Acumulador = Y1 - Y2 *)
        ST    Temp        (* Salva resultado intermediário em Temp *)
        MUL   Temp        (* Acumulador = Temp ^2*)
        ADD(  X1          (* Acc2 = X1, empilha soma *)
```

```
        SUB   X2           (* Acc2 = X1 - X2 *)
        ST    Temp         (* Salva em Temp *)
        MUL   Temp         (* Acc2 = Temp^2 *)
        )                  (* Executa soma suspensa *)
        ST    Temp         (* Salva em Temp *)
        CAL   SQRT(Temp)   (* Função raiz quadrada *)
        ST    TRAVEL       (* Estabelece saída da função *)
        GT    Tmax         (* Testa se maior que Tmax *)
        JMPC  ERR          (* Sim, vá para erro *)
        S     ENO          (* Set ENO *)
        RET                (* Retorno normal *)
ERR:
        RET                (* Retorno sem setar ENO *)
END_FUNCTION
```

Agora desenhe o equivalente em *Block Language*.

Leitura Complementar:

- Bonfatti, Monari, Sampieri, IEC1131-3 Prograrmming Methodology, CJ International, 1997.

Exercícios:

1. Escreva um bloco de função em IL para calcular a seguinte função de variáveis complexas:

   $z_{n+1} = z^2_n + p$ **, onde** $p = x + jy$

2. Dê três exemplos de situações onde você acha vantajoso utilizar a linguagem IL:

   a) _____

   b) _____

   c) _____

3. Comente a seguinte frase: "Não é vantagem se utilizar linguagens de muito baixo nível na programação de aplicações, porque o baixo nível de abstração resulta em programas ilegíveis e de alta complexidade".

4. Leia na Internet sobre a linguagem Step 5 da Siemens e a compare com a linguagem IL.

5. Escreva um programa que converta expressões na forma infixada para expressões na forma posfixada.

   Exemplo:

   Inorder: (a + b / c) * d

   Postorder: a b c / + d *

6. Escreva um programa em IL para calcular a expressão:

   Exp = X + A * (B+C) – (A * (B-C));