



Azionamenti e sistemi per l'automazione e la domotica

Luca Sani

Dipartimento di Sistemi Elettrici e Automazione

Università di Pisa

tel. 050 2217364

email luca.sani@dsea.unipi.it

The background of the slide features a large, faint, circular watermark of the University of Illinois seal. The seal contains the text "UNIVERSITY OF ILLINOIS" around the top and "1843" at the bottom, with a central emblem.

Linguaggio Instruction List (IL)

IEC 1131-3: Instruction List (IL)

Caratteristiche Principali:

- struttura simile ad un linguaggio assembler

Nel linguaggio Ladder si parlava di '**contatti**', nel IL si parla di '**variabili**'

- definito includendo le più diffuse istruzioni a basso livello dei PLC industriali attuali
- e' utilizzato insieme a linguaggi come il Sequential Function Chart (SFC)
- permette l'ottimizzazione del codice, nel caso si debbano codificare piccoli programmi

Caratteristiche accessorie del Linguaggio IL

- E' possibile definire un Function Block o una Funzione (con parametri)
- E' possibile richiamare un Function Block o una Funzione definita dall'utente
- E' possibile richiamare un Function Block o una Funzione predefinita

Università di Pisa - Dipartimento di
Sistemi Elettrici e Automazione

Regole di esecuzione

Un programma scritto in linguaggio IL viene eseguito valutando una riga di codice alla volta.

Il flusso di dati può andare sempre e solo dall'alto verso il basso, senza possibilità di inversione.

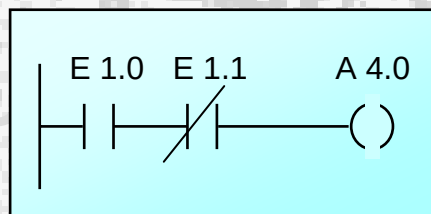
L'ordine di valutazione delle righe di codice è quello che procede dalla prima riga in alto verso l'ultima riga in basso.

Quando l'ultima riga viene valutata, si inizia nuovamente a valutare la prima riga (dopo aver aggiornato le uscite e letti gli ingressi).

Regole di esecuzione

Un rung dello schema ladder corrisponde ad una funzione logica.

Rung



Funzione logica

$$E\ 1.0 * \overline{E\ 1.1} = A\ 4.0$$

Simboli:

* AND

+ OR

— Negato

(

)

= risultato funzione

Struttura del Linguaggio IL

- il linguaggio IL traduce ogni funzione logica in un set di istruzioni.
- ogni istruzione e' scritta su una linea
- ogni istruzione e' composta da un operatore seguito da un operando
- ogni operazione fa riferimento ad un registro risultato chiamato "ACCU"
- ogni istruzione può essere "etichettata", per realizzare un Jump

Esempio di programma IL:

```
LD      Speed
GT      1000
JMPCN VOLTS_OK
LD      Volts
SUB     Volts
VOLTS_OK: LD      1
          ST      %Q75
```

Instruction List Codifica S7

Nel PLC Siemens il linguaggio di programmazione Instruction List è chiamato AWL

Ogni variabile è individuata dal seguente codice:

X A.B

X lettera che identifica il tipo di elemento
A.B indirizzo dell'elemento

Elemento	Simbolo	Esempio
INGRESSI	E	(E 0.4)
USCITE	A	(A 3.4)
VARIABILI INTERNE	M	(M 1.4)
TEMPORIZZATORI	T	(T 2.4)
CONTATORI	C	(C 0.4)

Set istruzioni del AWL

Operazioni logiche combinatorie di bit

- 1.2 U AND
- 1.3 UN AND negato
- 1.4 O OR
- 1.5 ON OR negato
- 1.6 X OR esclusivo
- 1.7 XN OR esclusivo negato
- 1.8 O AND prima di OR
- 1.9 U(AND con apertura parentesi
- 1.10 UN(AND negato con apertura parentesi
- 1.11 O(OR con apertura parentesi
- 1.12 ON(OR negato con apertura parentesi
- 1.13 X(OR esclusivo con apertura parentesi
- 1.14 XN(OR esclusivo negato con apertura parentesi
- 1.15) Chiusura parentesi
- 1.16 = Assegna
- 1.17 R Resetta
- 1.18 S Imposta
- 1.19 NOT Nega RLC
- 1.20 SET Imposta RLC (=1)
- 1.21 CLR Resetta RLC (=0)
- 1.22 SAVE Salva RLC nel registro BIE
- 1.23 FN Fronte di discesa
- 1.24 FP Fronte di salita

Set istruzioni del AWL

Operazioni di confronto

- 2.2 ? I Confronta numeri interi (a 16 bit)
- 2.3 ? D Confronta numeri interi (a 32 bit)
- 2.4 ? R Confronta numeri in virgola mobile (a 32 bit)

Operazioni di conversione

- 3.2 BTI Converti numero BCD in numero intero (a 16 bit)
- 3.3 ITB Converti numero intero (a 16 bit) in numero BCD
- 3.4 BTD Converti numero BCD in numero intero (a 32 bit)
- 3.5 ITD Converti numero intero (a 16 bit) in numero intero (a 32 bit)
- 3.6 DTB Converti numero intero (a 32 bit) in numero BCD
- 3.7 DTR Converti numero intero (a 32 bit) in numero in virgola mobile (a 32 bit, IEEE-FP)
- 3.8 INVI Complemento a 1 di numero intero (a 16 bit)
- 3.9 INVD Complemento a 1 di numero intero (a 32 bit)
- 3.10 NEGI Complemento a 2 di numero intero (a 16 bit)
- 3.11 NEGD Complemento a 2 di numero intero (a 32 bit)
- 3.12 NEGR Complemento a 2 di numero in virgola mobile (a 32 Bit, IEEE-FP)
- 3.13 TAW Cambia sequenza di byte in ACCU 1 (a 16 bit)
- 3.14 TAD Cambia sequenza di byte in ACCU 1 (a 32 bit)
- 3.15 RND Arrotonda al numero intero
- 3.16 TRUNC Arrotonda senza resto
- 3.17 RND+ Arrotonda al numero intero superiore (a 32 bit)
- 3.18 RND- Arrotonda al numero intero inferiore (a 32 bit)

Set istruzioni del AWL

Operazioni di conteggio

- 4.2 FR Abilita contatore
- 4.3 L Carica valore attuale di conteggio in ACCU 1 come numero intero
- 4.4 LC Carica valore attuale di conteggio in ACCU 1 come BCD
- 4.5 R Resetta contatore
- 4.6 S Imposta valore iniziale di conteggio
- 4.7 ZV Conta in avanti
- 4.8 ZR Conta all'indietro

5 Operazioni di blocchi di dati

- 5.2 AUF Apri blocco dati
- 5.3 TDB Scambia DB globale e DB di istanza
- 5.4 L DBLG Carica lunghezza del DB globale in ACCU 1
- 5.5 L DBNO Carica numero del DB globale in ACCU 1
- 5.6 L DILG Carica lunghezza del DB di istanza in ACCU 1
- 5.7 L DINO Carica numero del DB di istanza in ACCU 1

Set istruzioni del AWL

Operazioni di conteggio

- 4.2 FR Abilita contatore
- 4.3 L Carica valore attuale di conteggio in ACCU 1 come numero intero
- 4.4 LC Carica valore attuale di conteggio in ACCU 1 come BCD
- 4.5 R Resetta contatore
- 4.6 S Imposta valore iniziale di conteggio
- 4.7 ZV Conta in avanti
- 4.8 ZR Conta all'indietro

5 Operazioni di blocchi di dati

- 5.2 AUF Apri blocco dati
- 5.3 TDB Scambia DB globale e DB di istanza
- 5.4 L DBLG Carica lunghezza del DB globale in ACCU 1
- 5.5 L DBNO Carica numero del DB globale in ACCU 1
- 5.6 L DILG Carica lunghezza del DB di istanza in ACCU 1
- 5.7 L DINO Carica numero del DB di istanza in ACCU 1

Set istruzioni del AWL

Operazioni di salto

- 6.2 SPA Salto assoluto
- 6.3 SPL Salta alle etichette
- 6.4 SPB Salta se $RLC = 1$
- 6.5 SPBN Salta se $RLC = 0$
- 6.6 SPBB Salta se $RLC = 1$ con BIE
- 6.7 SPBNB Salta se $RLC = 0$ con BIE
- 6.8 SPBI Salta se $BIE = 1$
- 6.9 SPBIN Salta se $BIE = 0$
- 6.10 SPO Salta se $OV = 1$
- 6.11 SPS Salta se $OS = 1$
- 6.12 SPZ Salta se risultato = 0
- 6.13 SPN Salta se risultato diverso da zero
- 6.14 SPP Salta se risultato > 0
- 6.15 SPM Salta se risultato < 0
- 6.16 SPPZ Salta se risultato ≥ 0
- 6.17 SPMZ Salta se risultato ≤ 0
- 6.18 SPU Salta se operazione non ammessa
- 6.19 LOOP Loop di programma

Set istruzioni del AWL

Funzioni con numeri in virgola fissa

- 7.2 Valutazione dei bit nella parola di stato con operazioni con numeri in virgola fissa
- 7.3 +I Somma ACCU 1 e ACCU 2 come numeri interi (a 16 bit)
- 7.4 -I Sottrai ACCU 1 da ACCU 2 come numeri interi (a 16 bit)
- 7.5 *I Moltiplica ACCU 1 per ACCU 2 come numeri interi (a 16 bit)
- 7.6 /I Dividi ACCU 2 per ACCU 1 come numeri interi (a 16 bit)
- 7.7 + Somma costante di numero intero (a 16, 32 bit)
- 7.8 +D Somma ACCU 1 e ACCU 2 come numeri interi (a 32 bit)
- 7.9 -D Sottrai ACCU 1 da ACCU 2 come numeri interi (a 32 bit)
- 7.10 *D Moltiplica ACCU 1 per ACCU 2 come numeri interi (a 32 bit)
- 7.11 /D Dividi ACCU 2 per ACCU 1 come numeri interi (a 32 bit)
- 7.12 MOD Resto della divisione di numero intero (a 32 bit)

Set istruzioni del AWL

8 Funzioni con numeri in virgola mobile

8.2 Valutazione dei bit nella parola di stato con operazioni in virgola mobile.

8.3 Operazioni di base

8.3.1 +R Somma ACCU 1 e ACCU 2 come numeri in virgola mobile (a 32 bit)

8.3.2 -R Sottrai ACCU 1 da ACCU 2 come numeri in virgola mobile (a 32 bit)

8.3.3 *R Moltiplica ACCU 1 per ACCU 2 come numeri in virgola mobile (a 32 bit)

8.3.4 /R Dividi ACCU 2 per ACCU 1 come numeri in virgola mobile (a 32 bit)

8.3.5 ABS Valore assoluto di un numero in virgola mobile (a 32 bit, IEEE-FP)

8.4 Operazioni avanzate

8.4.1 SQR Formazione del quadrato di un numero in virgola mobile (a 32 bit)

8.4.2 SQRT Formazione della radice quadrata di un numero in virgola mobile (a 32 bit)

8.4.3 EXP Formazione del valore esponenziale di un numero in virgola mobile (a 32 bit)

8.4.4 LN Formazione del logaritmo naturale di un numero in virgola mobile (a 32 bit)

8.4.5 SIN Formazione del seno di un angolo come numero in virgola mobile (a 32 bit)

8.4.6 COS Formazione del coseno di un angolo come numero in virgola mobile (a 32 bit)

8.4.7 TAN Formazione della tangente di un angolo come numero in virgola mobile (a 32 bit)

8.4.8 ASIN Formazione dell'arcoseno di un numero in virgola mobile (a 32 bit)

8.4.9 ACOS Formazione dell'arcocoseno di un numero in virgola mobile (a 32 bit)

8.4.10 ATAN Formazione dell'arcotangente di un numero in virgola mobile (a 32 bit)

Set istruzioni del AWL

Operazioni di caricamento e trasferimento

- 9.2 L Carica
- 9.3 L STW Carica la parola di stato in ACCU 1
- 9.4 LAR1 Carica il registro di indirizzo 1 con il contenuto di ACCU 1
- 9.5 LAR1 <D> Carica il registro di indirizzo 1 con numero intero a 32 bit
- 9.6 LAR1 AR2 Carica registro di indirizzo 1 con contenuto del registro di indirizzo 2
- 9.7 LAR2 Carica il registro di indirizzo 2 con il contenuto di ACCU 1
- 9.8 LAR2 <D> Carica il registro di indirizzo 2 con numero intero a 32 bit
- 9.9 T Trasferisci
- 9.10 T STW Trasferisci ACCU 1 nella parola di stato
- 9.11 TAR cambia registro di indirizzo 1 con registro di indirizzo 2
- 9.12 TAR1 Trasferisci registro di indirizzo 1 in ACCU 1
- 9.13 TAR1 <D> Trasferisci registro di indirizzo 1 all'indirizzo destinazione (a 32 bit)
- 9.14 TAR1 AR2 Trasferisci registro di indirizzo 1 nel registro di indirizzo 2
- 9.15 TAR2 Trasferisci registro di indirizzo 2 in ACCU 1
- 9.16 TAR2 <D> Trasferisci registro di indirizzo 2 all'indirizzo di destinazione

Set istruzioni del AWL

Operazioni di controllo del programma

- 10.2 BE Fine blocco
- 10.3 BEA Fine blocco assoluto
- 10.4 BEB Fine blocco condizionato
- 10.5 CALL Richiamo di blocco
- 10.6 Richiamo di FB
- 10.7 Richiamo di FC
- 10.8 Richiamo di SFB
- 10.9 Richiamo di SFC
- 10.10 Richiamo di multiistanze
- 10.11 Richiamo di blocchi da una biblioteca
- 10.12 CC Richiamo condizionato di un blocco
- 10.13 UC Richiamo incondizionato di un blocco
- 10.14 Relè Master Control (MCR)
- 10.15 Avvertenze importanti sulle funzionalità MCR
- 10.16 MCR(Salva RLC nello stack MCR, inizio zona MCR
- 10.17)MCR Fine zona MCR
- 10.18 MCRA Attiva zona MCR
- 10.19 MCRD Disattiva zona MCR

Set istruzioni del AWL

Operazioni di scorrimento e rotazione

- 11.1.2 SSI Fai scorrere numero intero con segno (a 16 bit)
- 11.1.3 SSD Fai scorrere numero intero con segno (a 32 bit)
- 11.1.4 SLW Fai scorrere parola a sinistra (a 16 bit)
- 11.1.5 SRW Fai scorrere parola a destra (a 16 bit)
- 11.1.6 SLD Fai scorrere doppia parola a sinistra (a 32 bit)
- 11.1.7 SRD Fai scorrere doppia parola a destra (a 32 bit)
- 11.2.2 RLD Fai ruotare doppia parola a sinistra (a 32 bit)
- 11.2.3 RRD Fai ruotare doppia parola a destra (a 32 bit)
- 11.2.4 RLDA Fai ruotare ACCU 1 a sinistra tramite A1 (a 32 bit)
- 11.2.5 RRDA Fai ruotare ACCU 1 a destra tramite A1 (a 32 bit)

12 Operazioni di temporizzazione

- 12.2 Aree di memoria e componenti di un temporizzatore
- 12.3 FR Abilita temporizzatore
- 12.4 L Carica valore attuale di conteggio in ACCU 1 come numero intero
- 12.5 LC Carica valore attuale di conteggio in ACCU 1 come BCD
- 12.6 R Resetta temporizzatore
- 12.7 SI Avvia temporizzatore come impulso
- 12.8 SV Avvia temporizzatore come impulso prolungato
- 12.9 SE Avvia temporizzatore come ritardo all'inserzione
- 12.10 SS Avvia temporizzatore come ritardo all'inserzione con memoria
- 12.11 SA Avvia temporizzatore come ritardo alla disinserzione

Set istruzioni del AWL

Operazioni logiche a parola

- 13.2 UW AND a parola (a 16 bit)
- 13.3 OW OR a parola (a 16 bit)
- 13.4 XOW R esclusivo a parola (a 16 bit)
- 13.5 UD AND a doppia parola (a 32 bit)
- 13.6 OD OR a doppia parola (a 32 bit)
- 13.7 XOD OR esclusivo a doppia parola (a 32 bit)

Operazioni per il funzionamento dell'accumulatore

- 14.2 TAK Scambia ACCU 1 con ACCU 2
- 14.3 PUSH CPU con due accumulatori
- 14.4 PUSH CPU con quattro accumulatori
- 14.5 POP CPU con due accumulatori
- 14.6 POP CPU con quattro accumulatori
- 14.7 ENT Immetti stack accumulatore
- 14.8 LEAVE Esci da stack accumulatore
- 14.9 DEC Decrementa ACCU 1
- 14.10 INC Incrementa ACCU 1
- 14.11 +AR1 Somma ACCU 1 al registro di indirizzo 1
- 14.12 +AR2 Somma ACCU 1 al registro di indirizzo 2
- 14.13 BLD Comando di visualizzazione del programma (NOP)
- 14.14 NOP 0 Nessuna operazione 0
- 14.15 NOP 1 Nessuna operazione 1

Sommario delle operazioni logiche combinatorie di bit

Le operazioni logiche combinatorie a bit operano con le due cifre: 1 e 0.

Le operazioni logiche combinatorie a bit interpretano gli stati di segnale di 1 e 0, e li combinano secondo la logica booleana per eseguire una varietà di funzioni.

Queste combinazioni producono un risultato di 1 o 0 che è chiamato “risultato logico combinatorio” (RLC).

Per le combinazioni logiche con operandi a bit vi sono le seguenti operazioni di base:

Simbolo	Operazione logica
U	AND
UN	AND negato
O	OR
ON	OR negato
X	OR esclusivo
XN	OR esclusivo negato

U AND

Formato

U <var>

con var E, A, M, L, D, T, Z

U interroga la variabile var sullo stato di segnale "1", e combina l'esito dell'interrogazione (1 se var =1, 0 se var =0) con RLC mediante una AND.

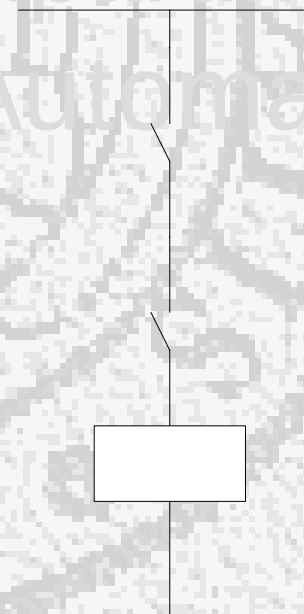
Funzione logica

$$E\ 1.0 * E\ 1.1 = A\ 4.0$$

U E 1.0

U E 1.1

= A 4.0



= Assegna

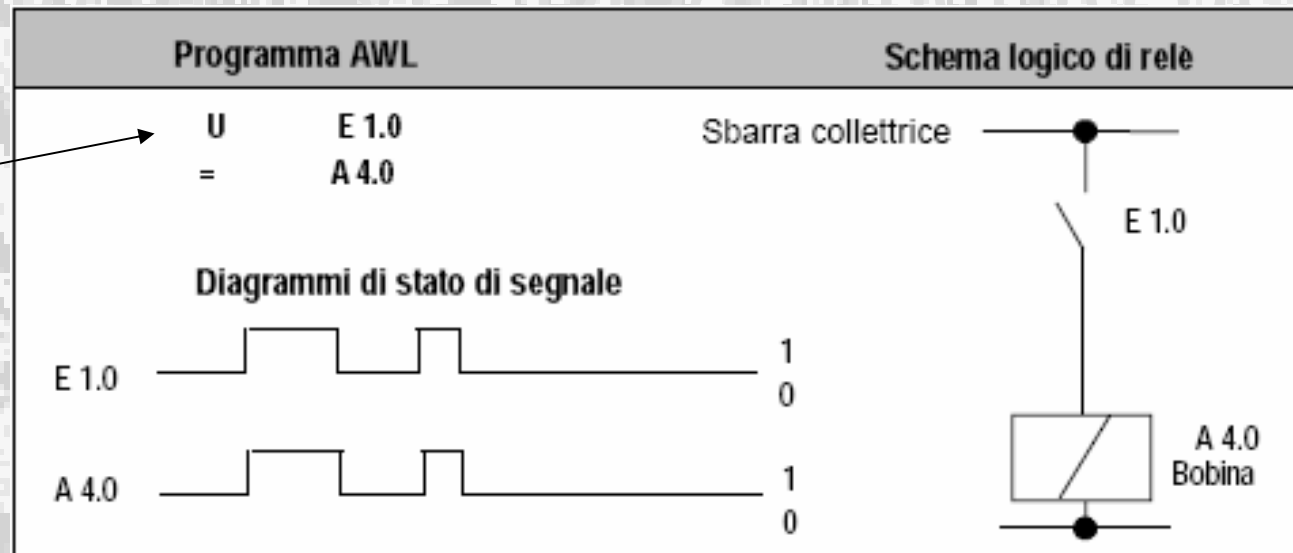
Formato

= <var>

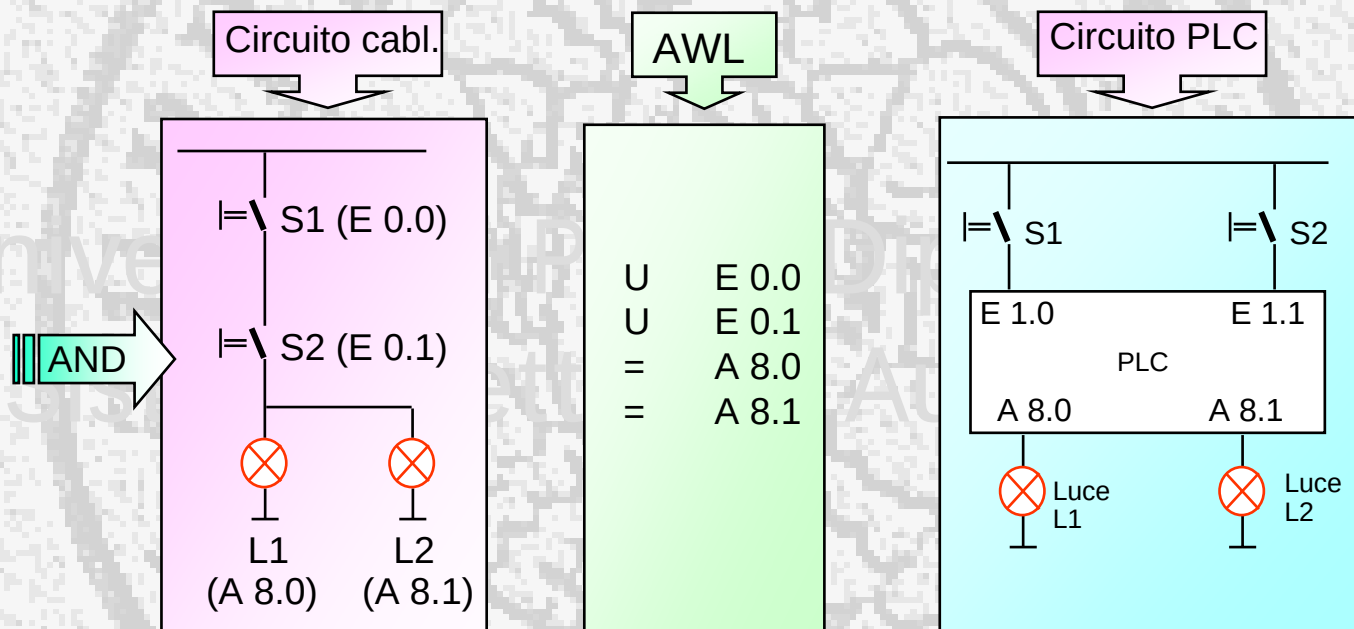
con var E, A, M, L, D

= assegna alla variabile var il valore del RLC.

Se non ci sono
funzioni logiche
l'operatore U
assume il
significato di start



Operazione logica binaria: AND



$$E 0.0 * E 0.1 = A 8.0$$

$$E 0.0 * E 0.1 = A 8.1$$

UN AND negato

Formato

UN <var>

con var E, A, M, L, D, T, Z

U interroga la variabile var sullo stato di segnale "0", e combina l'esito dell'interrogazione (1 se var =0, 0 se var =1) con RLC mediante una AND.

A 4.0 si attiva se sono premuti entrambi i pulsanti

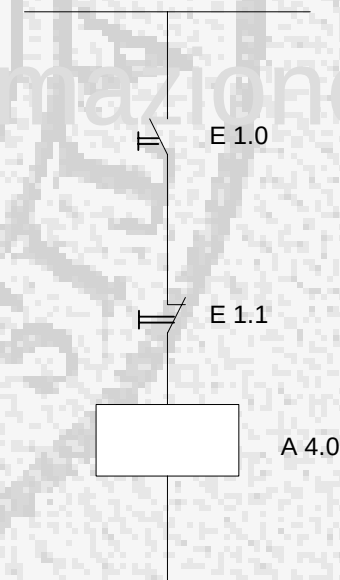
Funzione logica

$$E\ 1.0 * \overline{E\ 1.1} = A\ 4.0$$

U E 1.0

UN E 1.1

= A 4.0

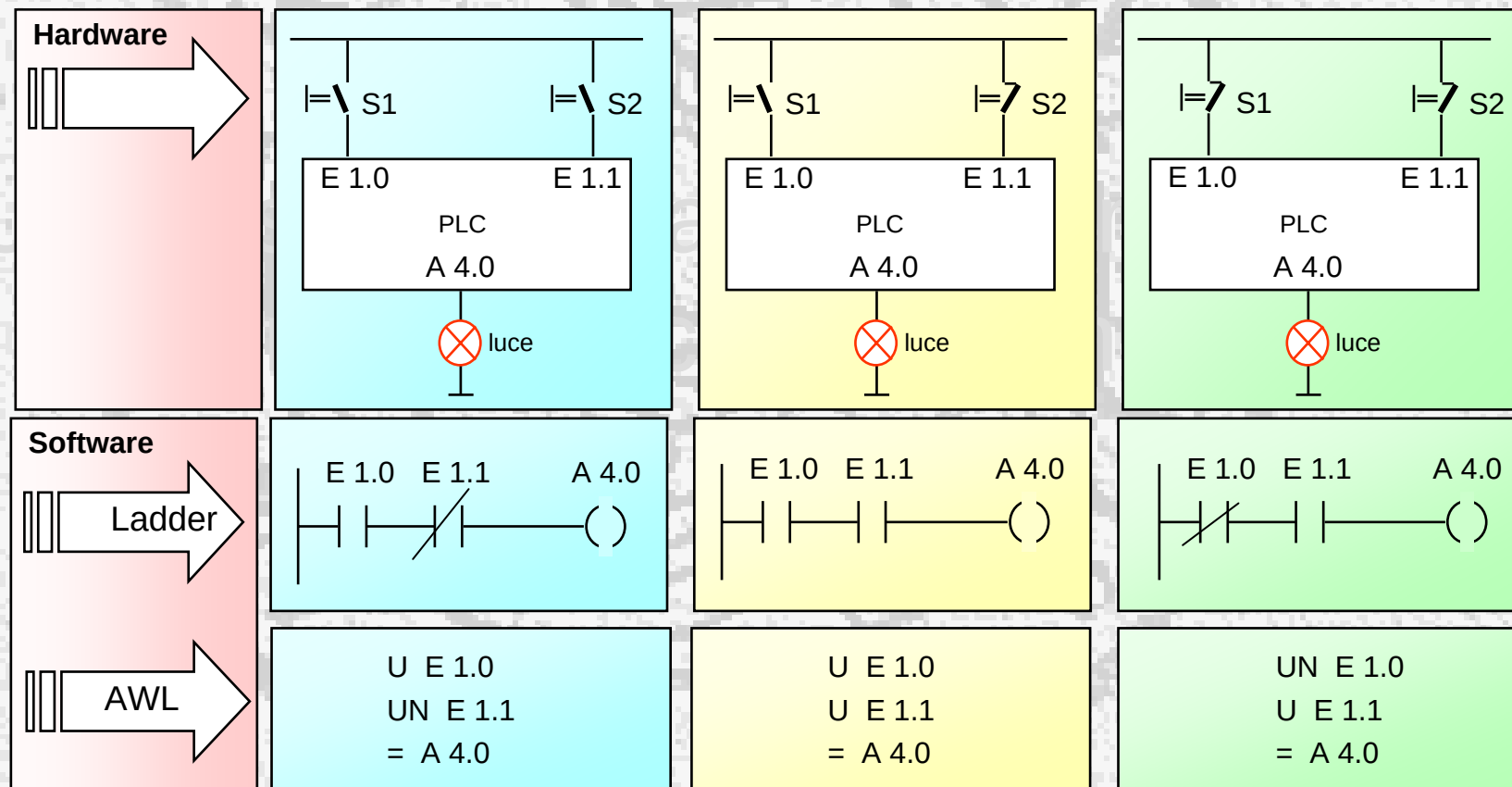


Risultato delle operazioni binarie. Esempi

		Esempio 1			Esempio 2			Esempio 3		
		Stato del segnale	Risultato dell'Interrog.	RLC	Stato del segnale	Risultato dell'Interrog.	RLC	Stato del segnale	Risultato dell'Interrog.	RLC
U	E 1.0	0	0	0	1	1	1	1	1	1
UN	E 1.1	0	1	0	1	0	0	0	1	1
U	M 4.0	0	0	0	1	1	0	1	1	1
=	A 8.0	0			0			1		
=	A 8.1	0			0			1		

Esercizio

In tutti e tre i casi la luce deve accendersi solo quando S1 è attivato e S2 non è attivato!



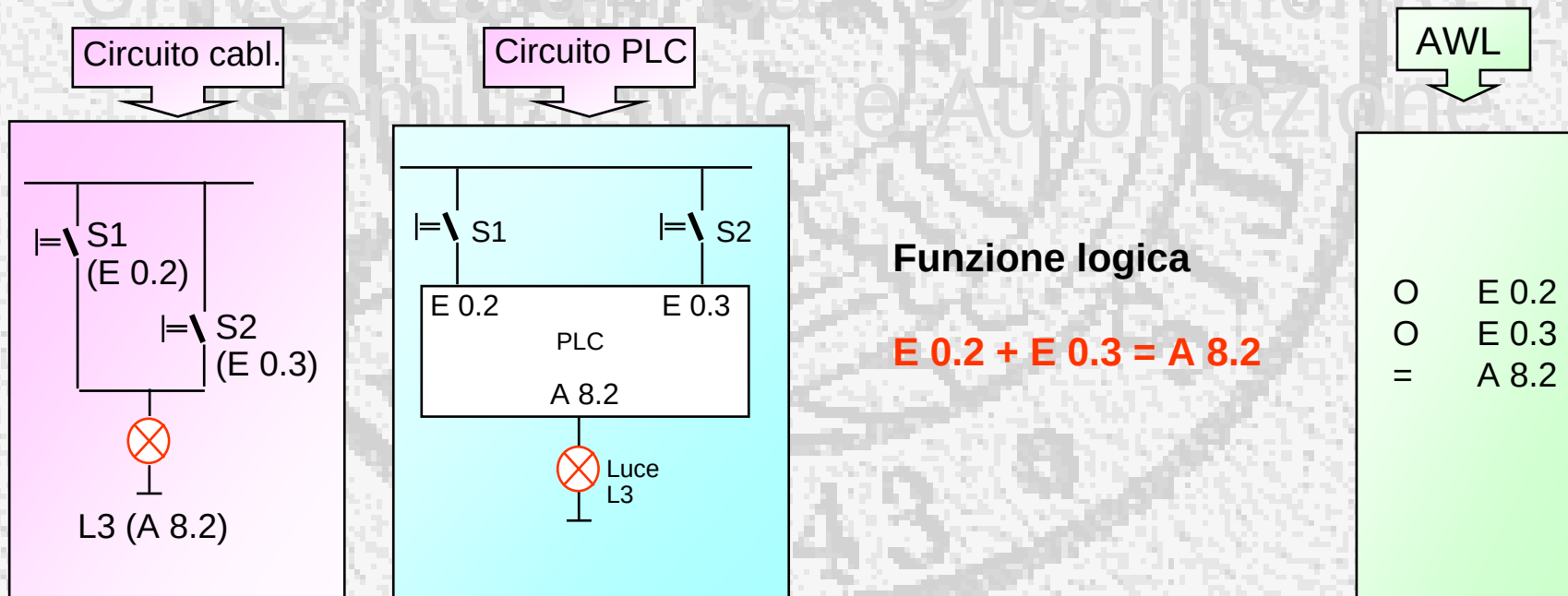
ON OR

Formato

O <var>

con var E, A, M, L, D, T, Z

O interroga la variabile var sullo stato di segnale "1", e combina l'esito dell'interrogazione (1 se var =1, 0 se var =0) con RLC mediante una OR.



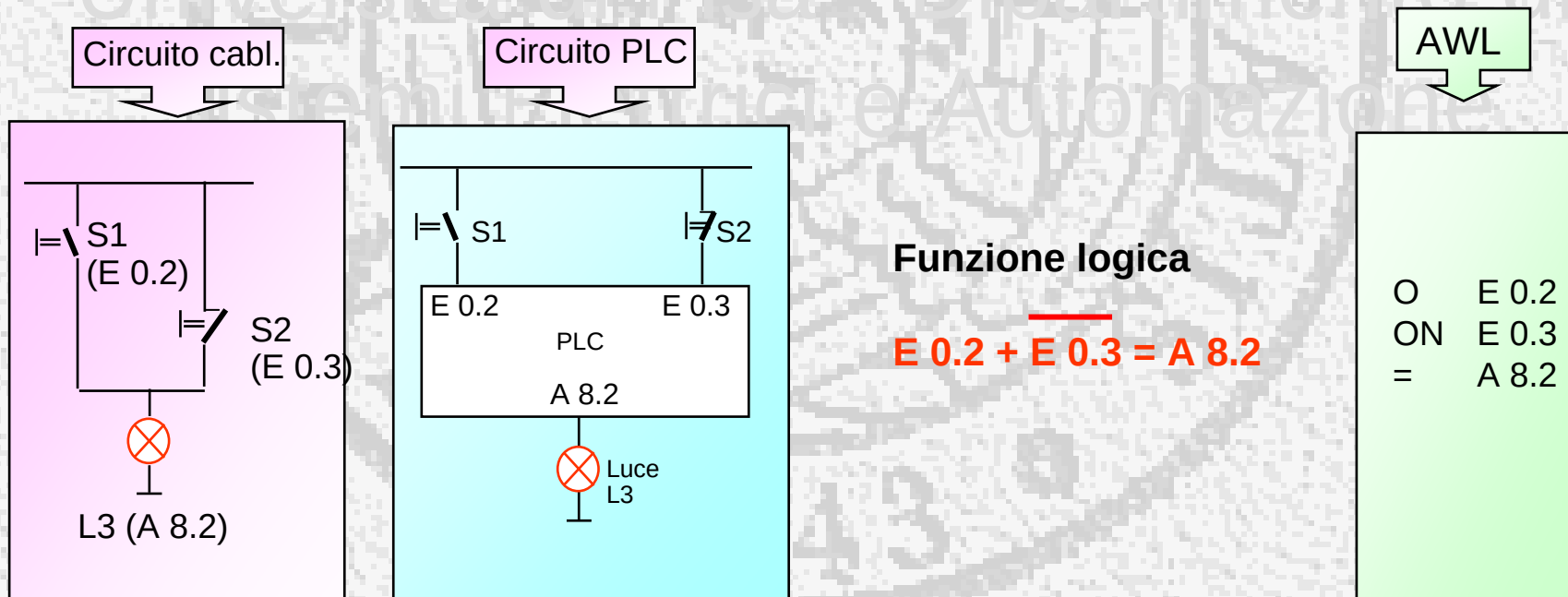
ON OR Negato

Formato

ON <var>

con var E, A, M, L, D, T, Z

ON interroga la variabile var sullo stato di segnale "0", e combina l'esito dell'interrogazione (1 se var =0, 0 se var =1) con RLC mediante una OR.



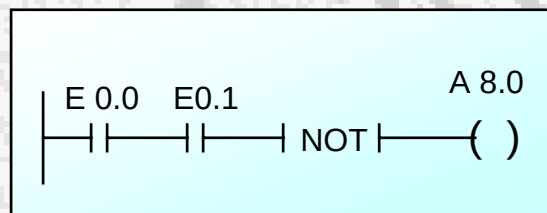
NOT

Formato

NOT

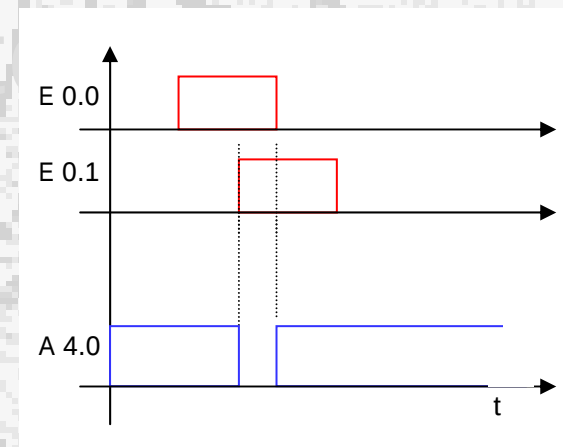
NOT nega il risultato logico combinatorio (RLC).

Ladder



AWL

U E 0.0
U E 0.1
NOT
= A 8.0



Uso delle parentesi

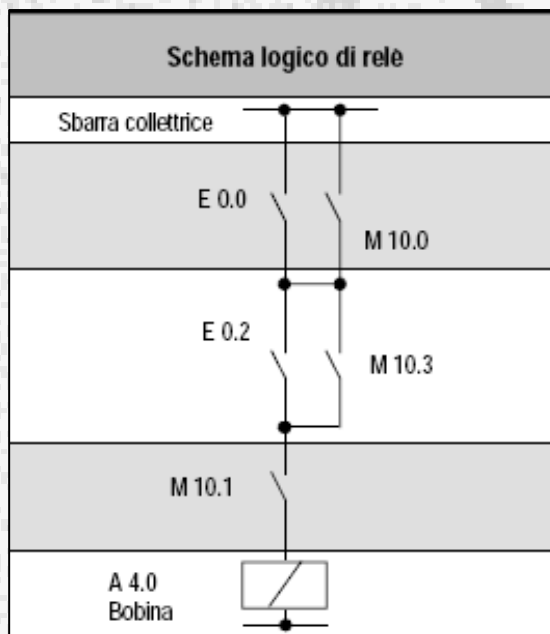
Si possono adoperare le seguenti operazioni per combinare espressioni a parentesi:

Simbolo	Operazione logica
U(	AND con diramazione
UN(	AND negato con apertura parentesi
O(	OR con apertura parentesi
ON(	OR negato con apertura parentesi
X(	OR esclusivo con apertura parentesi
XN(	OR esclusivo negato con apertura parentesi
)	Chiusura parentesi

Il segno di parentesi aperta è sempre preceduto dall'operatore logico di cui il blocco tra parentesi fa parte. Sono ammessi fino a 7 livelli di parentesi annidate.

Uso delle parentesi

Si considera il seguente esempio



La funzione logica associata è:

$$(E\ 0.0 + M\ 10.0) * (E\ 0.2 + M\ 10.3) * M\ 10.1 = A\ 4.0$$

Il codice AWL corrispondente è:

```

U(
O E0.0
O M 10.0
)
U(
O E0.2
O M10.3
)
U M10.1
= A 4.0
  
```

Uso delle parentesi

Si considera il seguente esempio

La funzione logica associata è:

$$((E\ 0.0 * M\ 10.0) + (E\ 0.2 + M\ 10.3)) * M\ 10.1 = A\ 4.0$$

Il codice AWL corrispondente è:

```
U(  
O(  
O E0.0  
O M 10.0  
)  
O(  
O E0.2  
O M10.3  
)  
)
```

```
U M10.1  
= A 4.0
```

Uso delle parentesi

Si considera il seguente esempio

La funzione logica associata è:

$$\overline{E\ 0.0} * (((\overline{E\ 0.1} + \overline{E\ 0.2}) * E\ 0.3) + E\ 0.4) * E\ 0.5 = A\ 4.0$$

Il codice AWL corrispondente è:

UN E 0.0

U(

O(

UN(

O E 0.1

O E 0.2

)

U E 0.3

)

O E 0.4

)

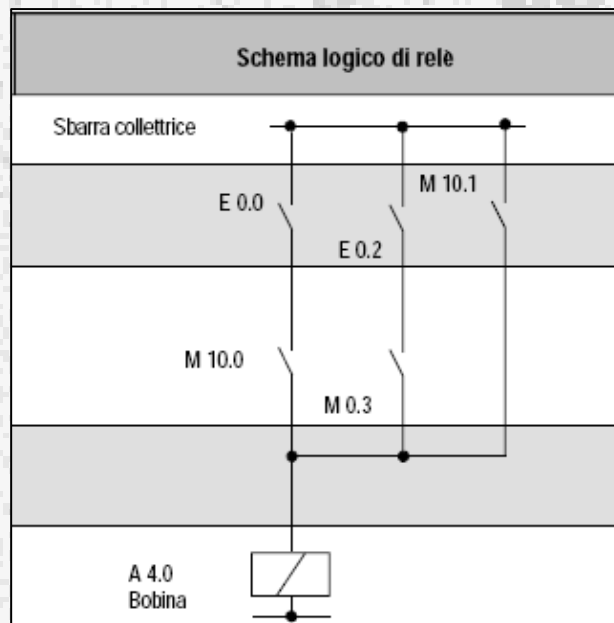
U M10.1

U E 0.5

A 4.0

Uso delle parentesi

Se la relazione logica è univoca le parentesi si possono omettere



La funzione logica associata è:

$$E\ 0.0 * M\ 10.0 + E\ 0.2 * M\ 0.3 + M\ 10.1 = A\ 4.0$$

La regola di precedenza delle operazioni (prima i prodotti (AND) poi le somme (OR) permette di evitare l'uso delle parentesi. Il codice AWL è:

```

U E0.0
U M10.0
O
U E 0.2
U M 0.3
O M10.1
= A 4.0

```

X XOR e XN XOR negato

Formato

X <var> con var E, A, M, L, D, T, Z

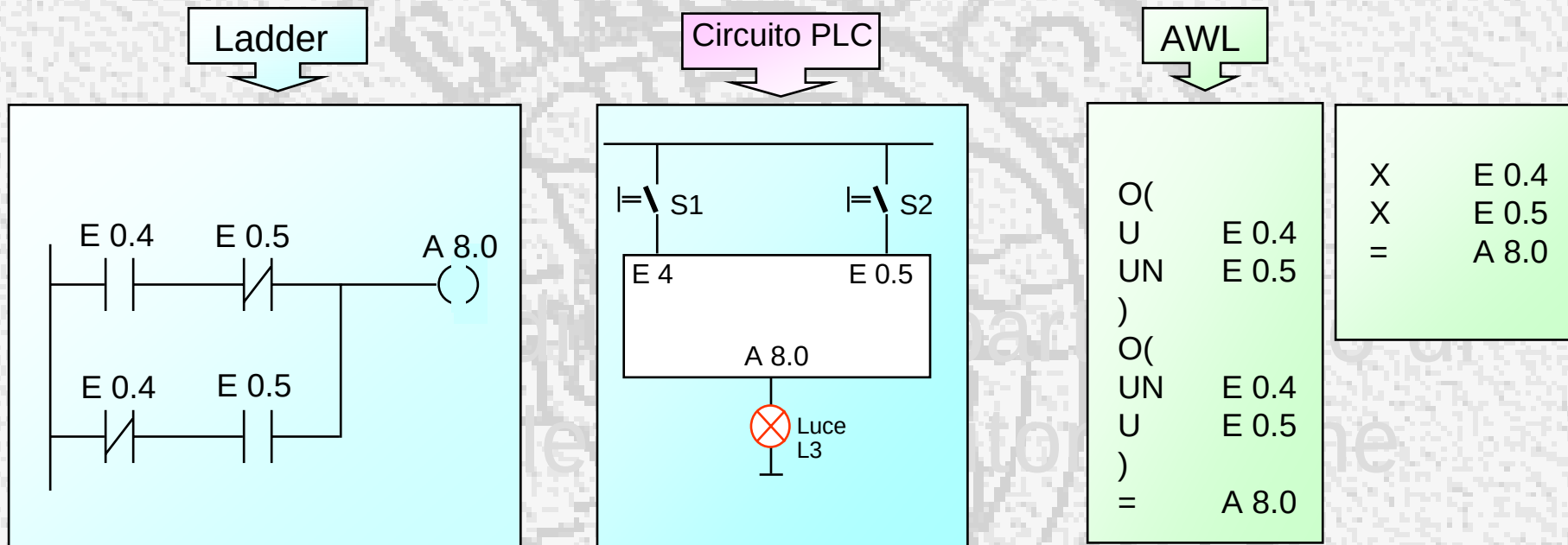
X interroga la variabile var sullo stato di segnale "1", e combina l'esito dell'interrogazione (1 se var =1, 0 se var =0) con RLC mediante una XOR.

Formato

XN <var> con var E, A, M, L, D, T, Z

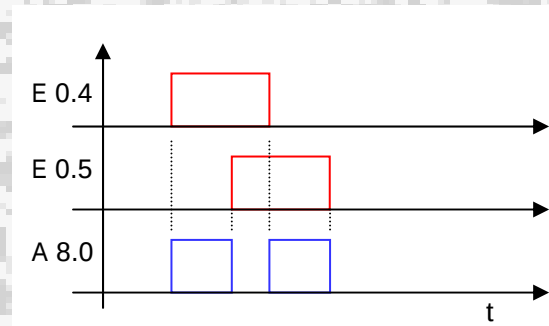
XN interroga la variabile var sullo stato di segnale "0", e combina l'esito dell'interrogazione (1 se var =0, 0 se var =1) con RLC mediante una XOR.

Operazioni logiche binarie: OR Esclusivo (XOR)



Funzione logica

$$(\overline{E 0.4} * E 0.5) + (E 0.4 * \overline{E 0.5}) = A 8.0$$



S SET, R RESET

Formato

S <var> con var E, A, M, L, D, T, Z

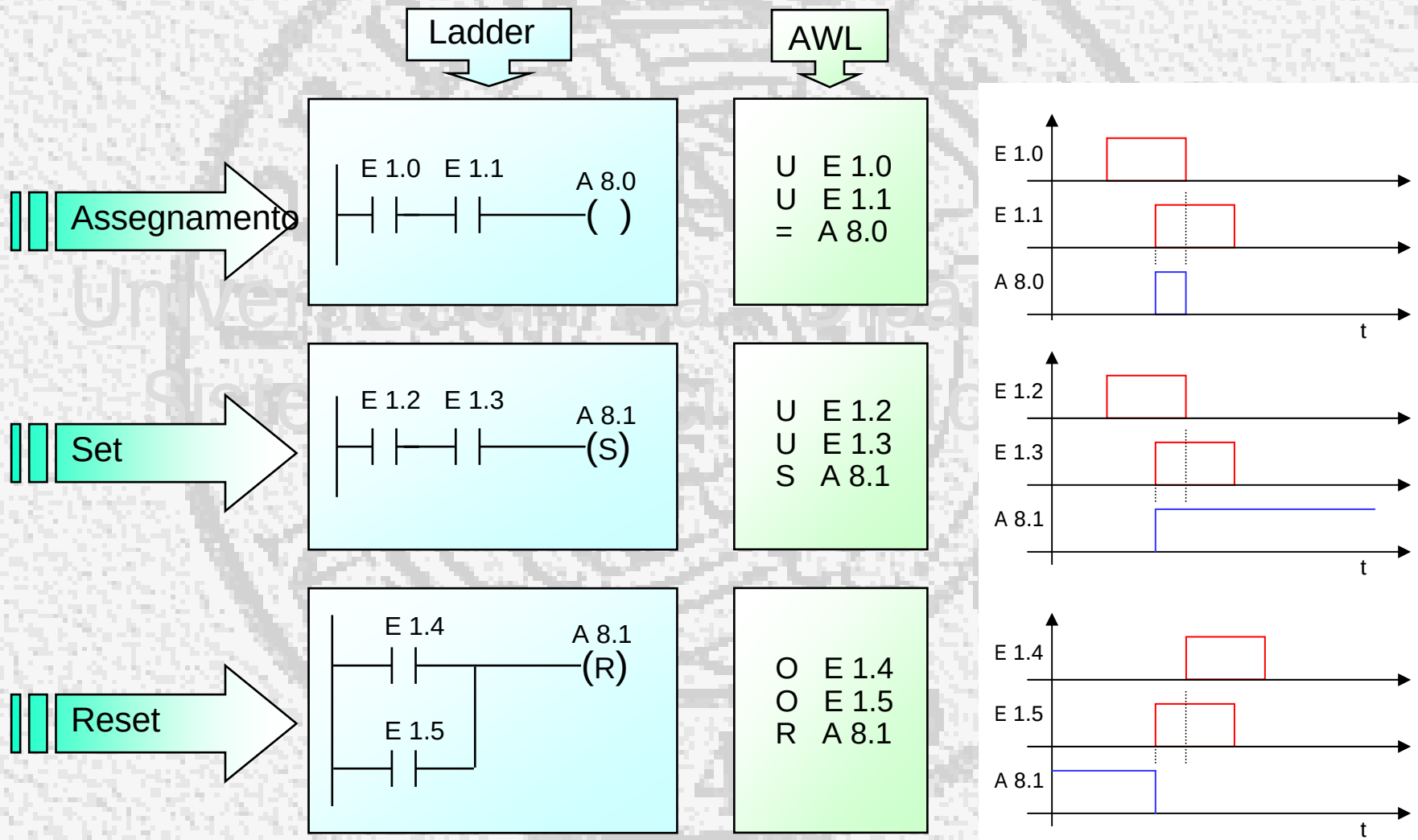
S (Imposta) scrive il valore "1" nella variabile var, se RLC = 1

Formato

R <var> con var E, A, M, L, D, T, Z

R (Imposta) scrive il valore "0" nella variabile var, se RLC = 1

Assegnamento, Set, Reset



FP FRONTE DI SALITA, FN FRONTE DI DISCESA

Formato

FS <var> con bit E, A, M, L, D, T, Z

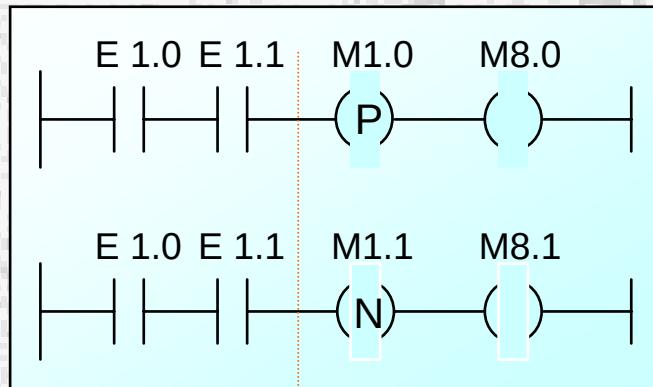
FS <var> (fronte di salita) rileva un fronte di salita quando RLC cambia da "0" a "1", e indica tale transizione con $RLC = 1$. Nella variabile var è memorizzato lo stato del RLC del passo precedente.

Formato

FN <var> con bit E, A, M, L, D, T, Z

FN <var> (fronte di discesa) rileva un fronte di discesa quando RLC cambia da "1" a "0", e indica tale transizione con $RLC = 1$. Nella variabile var è memorizzato lo stato del RLC del passo precedente.

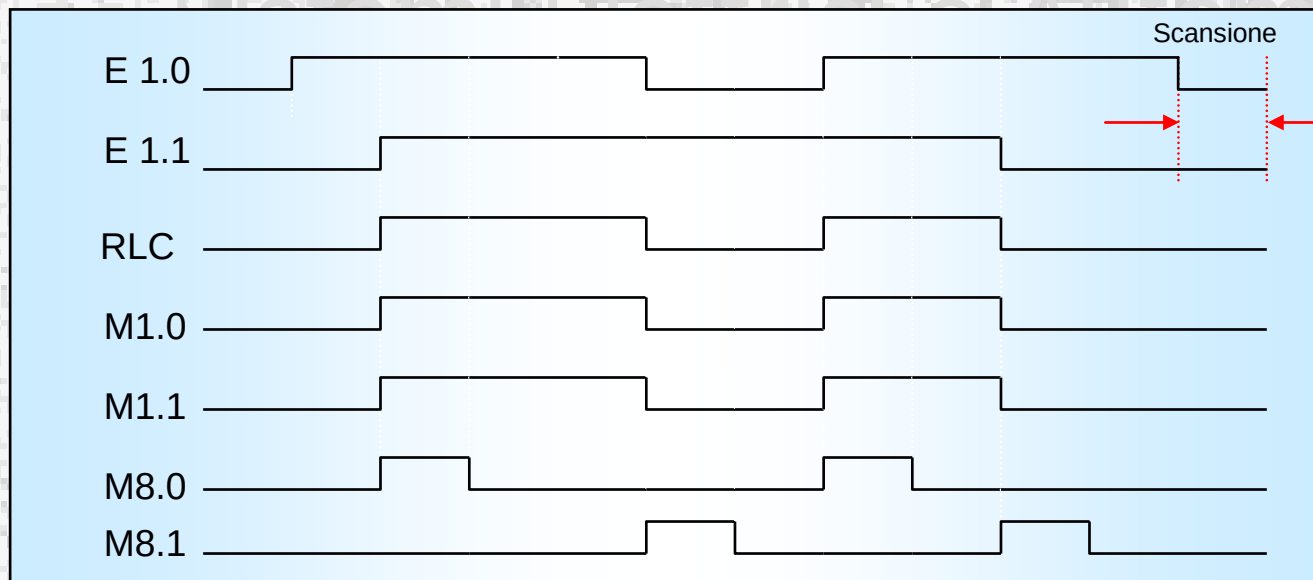
Rilevazione del fronte



UE 1.0
UE 1.1
FP M 1.0
= M 8.0

UE 1.0
UE 1.1
FN M 1.1
= M 8.1

RLC = Risultato Logico Combinatorio



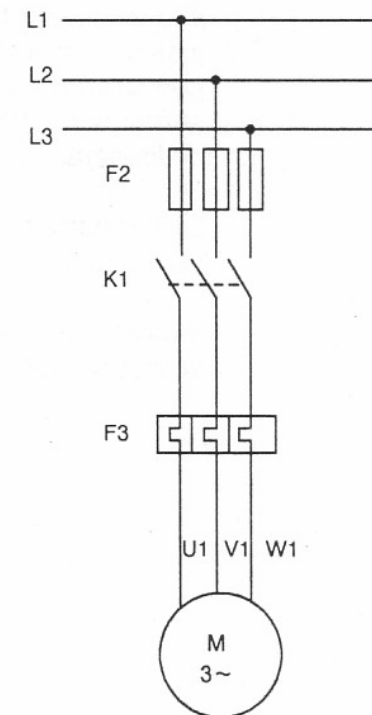
Esempio 1. Avviamento/Arresto di un motore

Descrizione del problema

Un motore asincrono viene inserito premendo il pulsante S1 (pulsante di marcia) e disinserito con il pulsante S0 (pulsante di arresto).

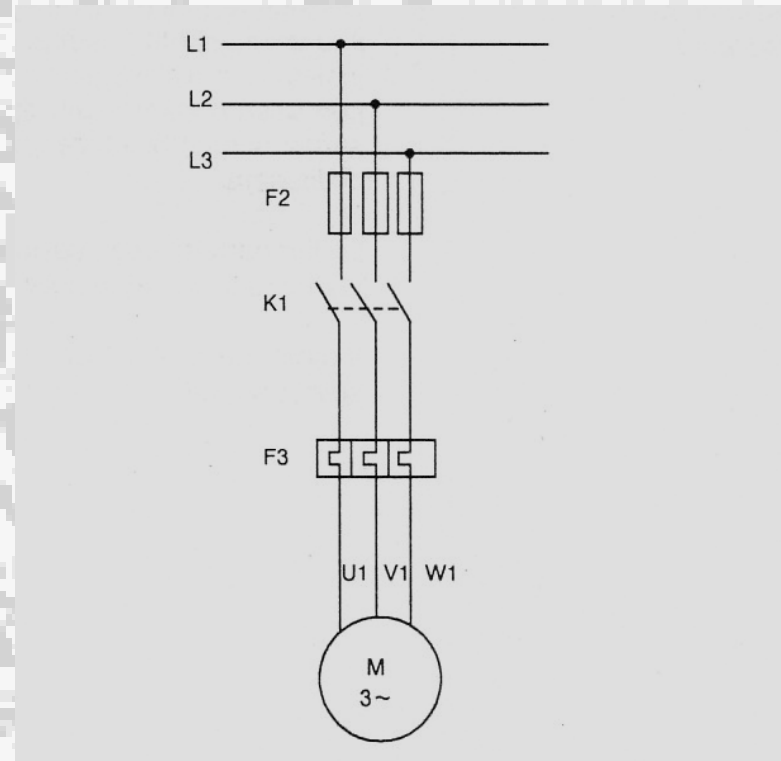
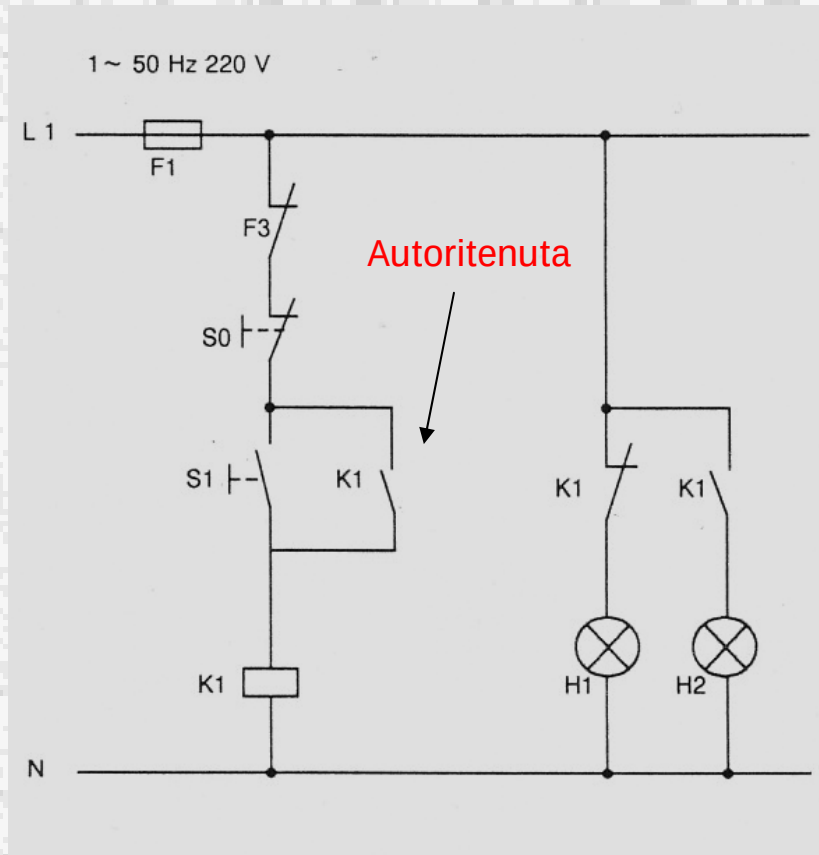
Le lampade di segnalazione H1 e H2 visualizzano lo stato di attività del motore.

Il motore è protetto con un interruttore termico di sovracorrente e con fusibili per il corto circuito.



Esempio 1. Avviamento/Arresto di un motore

In logica cablata il circuito di controllo è:



K1 contatti ausiliari del contattore

Esempio 1. Avviamento/Arresto di un motore

Si definiscono gli ingressi e si associano ai corrispondenti ingressi del PLC.

Simbolo	Variabile PLC	Descrizione
S0	E 0.1	OFF Pulsante N.A.
S1	E 0.2	ON Pulsante N.A.

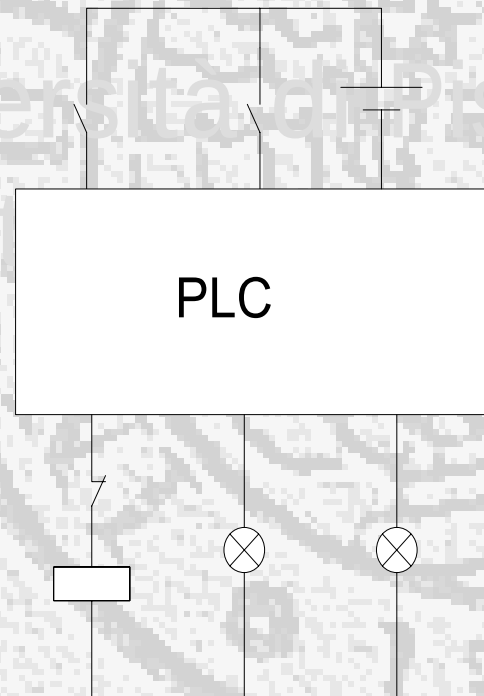
Esempio 1. Avviamento/Arresto di un motore

Si definiscono le uscite e si associano alle corrispondenti uscite del PLC.

Simbolo	Variabile PLC	Descrizione
K1	A 2.0	Bobina contattore motore
H1	A 2.1	Segnalazione motore fermo
H2	A 2.2	Segnalazione motore in marcia

Esempio 1. Avviamento/Arresto di un motore

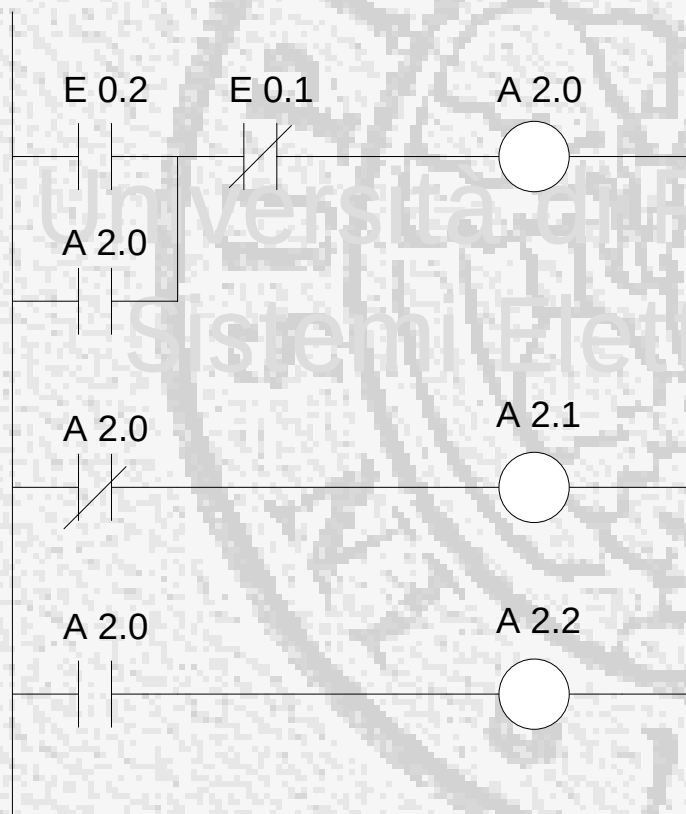
Lo schema di collegamento al PLC è il seguente:



Per motivi di sicurezza il contatto normalmente chiuso dell'interruttore di sovracorrente F3 deve essere collegato a monte della bobina del contattore K1. Se si attiva l'interruttore di sovracorrente, la bobina del contattore K1 viene disinserita direttamente via hardware senza l'intervento del PLC. L'interruttore termico di sovracorrente deve essere dotato di un dispositivo per il ripristino manuale per evitare un successivo inserimento del contattore K1 dovuto al suo ripristino automatico.

Esempio 1. Avviamento/Arresto di un motore

Il programma ladder e le funzioni logiche associate sono:



$$(E\ 0.2 + A\ 2.0) * \overline{E\ 0.1} = A\ 2.0$$

$$\overline{A\ 2.0} = A\ 2.1$$

$$A\ 2.0 = A\ 2.2$$

Il codice AWL è:

U(	UN A 2.0
O E 0.2	= A 2.1
O A 2.0	U A 2.0
)	= A 2.2
UN E 0.1	
= A 2.0	

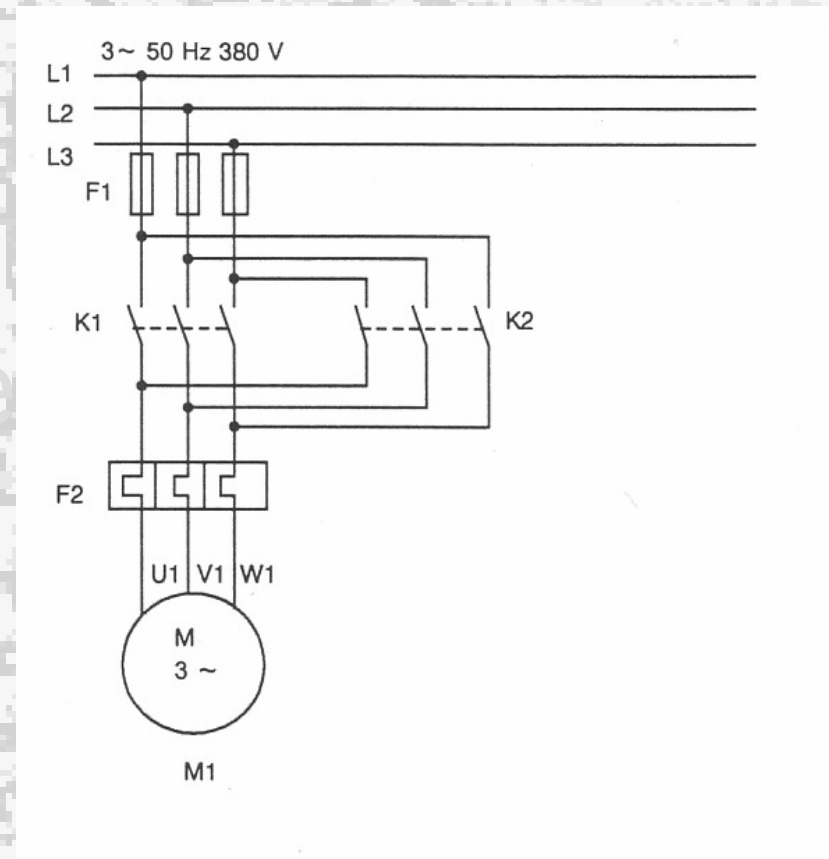
Esempio 2. Circuito invertitore per motore trifase

Descrizione del problema

Un motore asincrono viene inserito premendo il pulsante S1 per il funzionamento “marcia avanti”, S2 per “marcia indietro”, e disinserito con il pulsante S0.

Le lampade di segnalazione H1, H2 e H3 visualizzano lo stato di attività del motore.

Il motore è protetto con un interruttore termico di sovracorrente e con fusibili per il corto circuito.



Esempio 2. Circuito invertitore per motore trifase

Si definiscono gli ingressi e si associano ai corrispondenti ingressi del PLC.

Simbolo	Variabile PLC	Descrizione
S0	E 0.1	OFF Pulsante N.A.
S1	E 0.2	Pulsante Marcia avanti N.A.
S2	E 0.3	Pulsante Marcia indietro N.A.

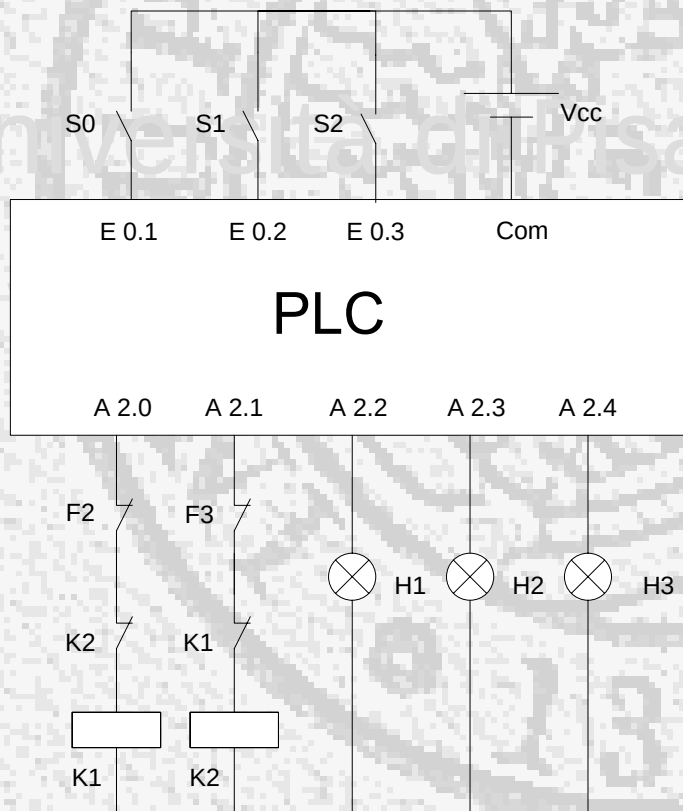
Esempio 2. Circuito invertitore per motore trifase

Si definiscono le uscite e si associano alle corrispondenti uscite del PLC.

Simbolo	Variabile PLC	Descrizione
K1	A 2.0	Bobina contattore funz. 'marcia avanti'
K2	A 2.1	Bobina contattore funz. 'marcia indietro'
H1	A 2.2	Segnalazione motore fermo
H2	A 2.3	Segnalazione motore in 'marcia avanti'
H3	A 2.4	Segnalazione motore in 'marcia indietro'

Esempio 2. Circuito invertitore per motore trifase

Lo schema di collegamento al PLC è il seguente:

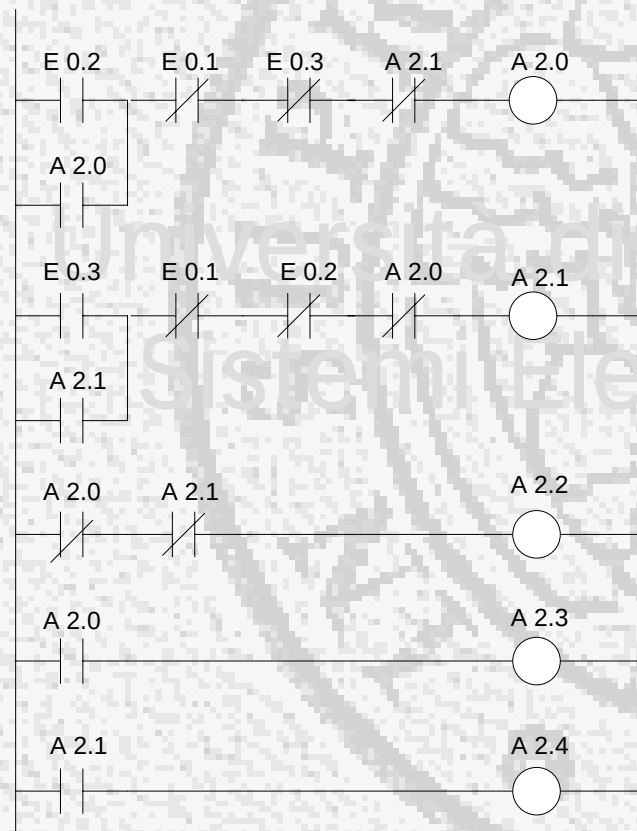


I contatti dell'interruttore termico devono essere collegati sulle linee di alimentazione dei due contattori.

Inoltre un contattore non deve poter essere inserito fintanto che l'altro non è completamente disinserito. La concomitanza di entrambi i comandi di eccitazione potrebbe verificarsi a causa delle velocissime reazioni di comando elettronico rispetto alle reazioni delle parti meccaniche in movimento.

Esempio 2. Circuito invertitore per motore trifase

Il programma ladder e le funzioni logiche associate sono:



$$(E\ 0.2 + A\ 2.0) * E\ 0.1 * E\ 0.3 * A\ 2.1 = A\ 2.0$$

$$(E\ 0.3 + A\ 2.1) * E\ 0.1 * E\ 0.2 * A\ 2.0 = A\ 2.1$$

$$A\ 2.0 * A\ 2.1 = A\ 2.2$$

$$A\ 2.0 = A\ 2.3$$

$$A\ 2.1 = A\ 2.4$$

Il codice AWL è:

U(	= A 2.0	UN A 2.0	U A 2.1
O E 0.2	U(	= A 2.1	= A 2.4
O A 2.0	O E 0.3	UN A 2.0	
)	O A 2.1	UN A 2.1	
UN E 0.1	)	= A 2.2	
UN E 0.3	UN E 0.1	U A 2.0	
UN A 2.1	UN E 0.2	= A 2.3	

Operazioni di caricamento e trasferimento

Le operazioni L (Carica) e T (Trasferisci) abilitano l'utente a programmare uno scambio di informazioni tra unità di ingresso e uscita e aree di memoria oppure tra aree di memoria.

La CPU esegue queste operazioni in ogni ciclo di scansione come operazioni incondizionate, vale a dire che queste operazioni non sono influenzate dal risultato logico combinatorio di un'istruzione.

L CARICA

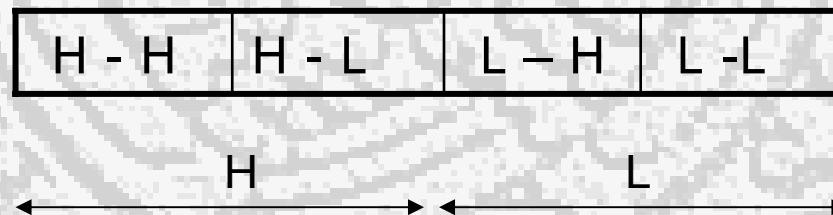
Formato

L <var>

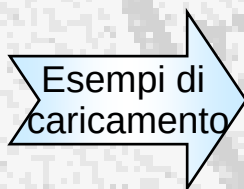
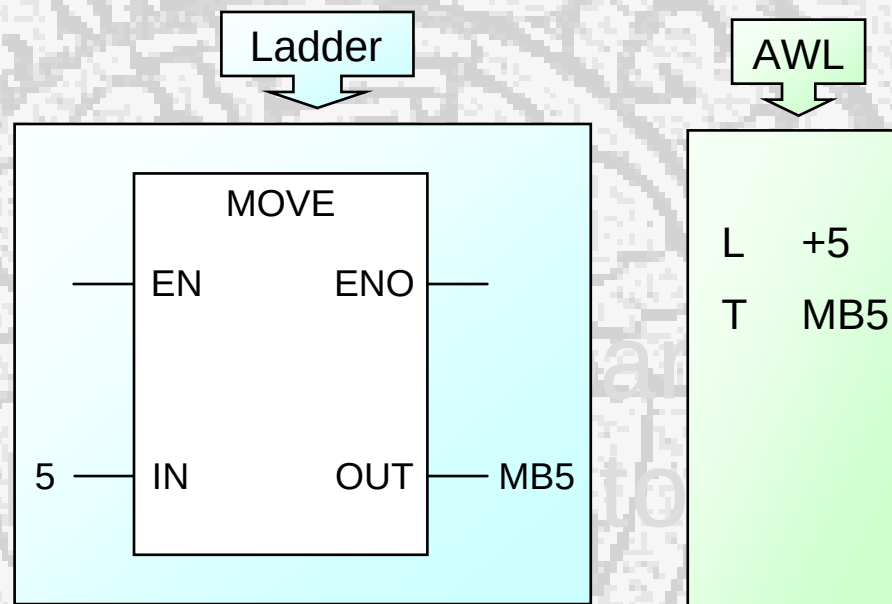
tipo di dato byte, word, dword

L carica nell'accumulatore 1 il byte, la parola o la doppia parola contenuta in var, dopo che il contenuto di ACCU 1 è stato memorizzato in ACCU 2, ed ACCU 1 è stato resettato a 0.

ACCU può essere suddiviso in 4 byte così identificati:

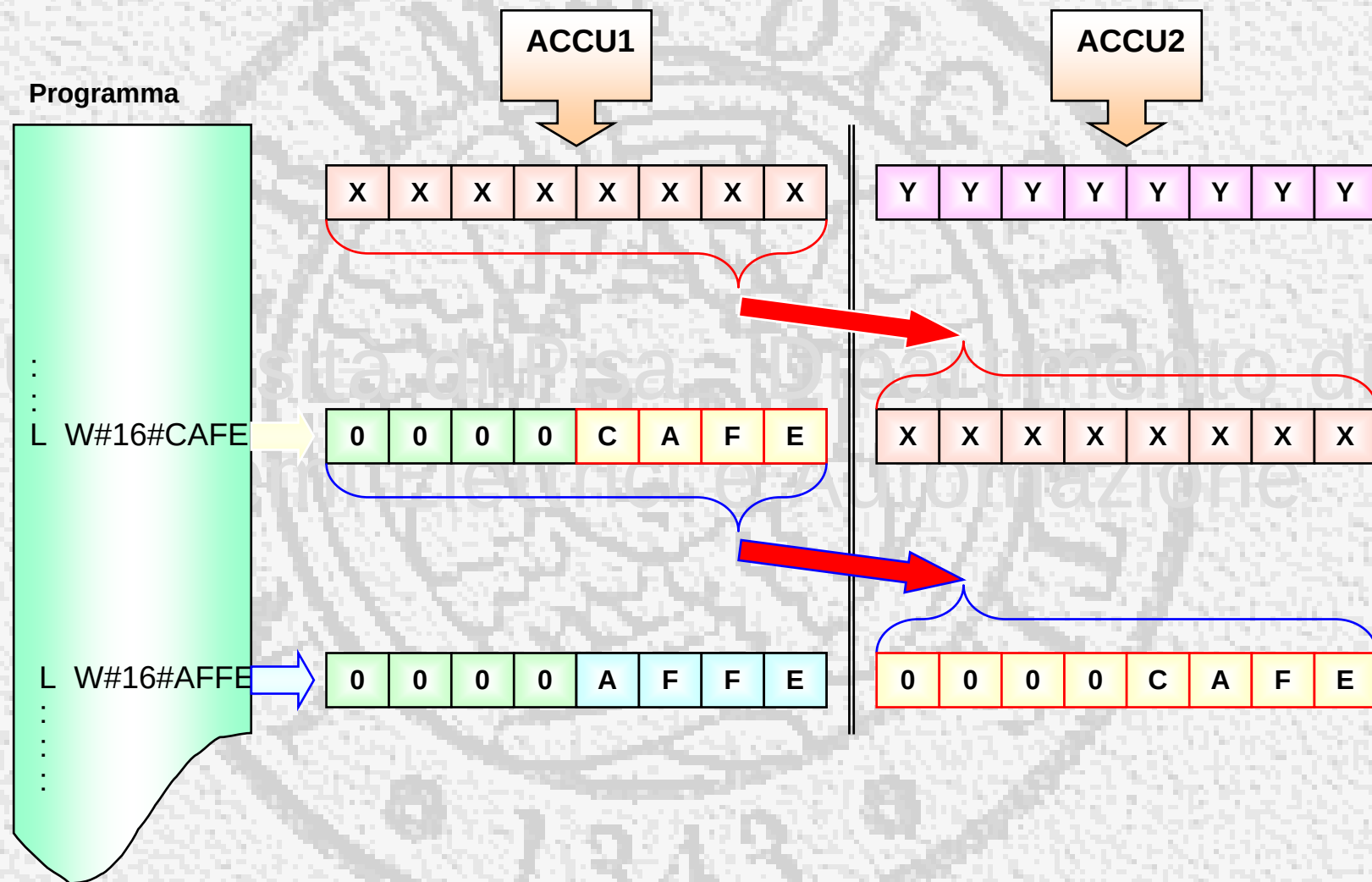


Caricamento e Trasferimento Dati

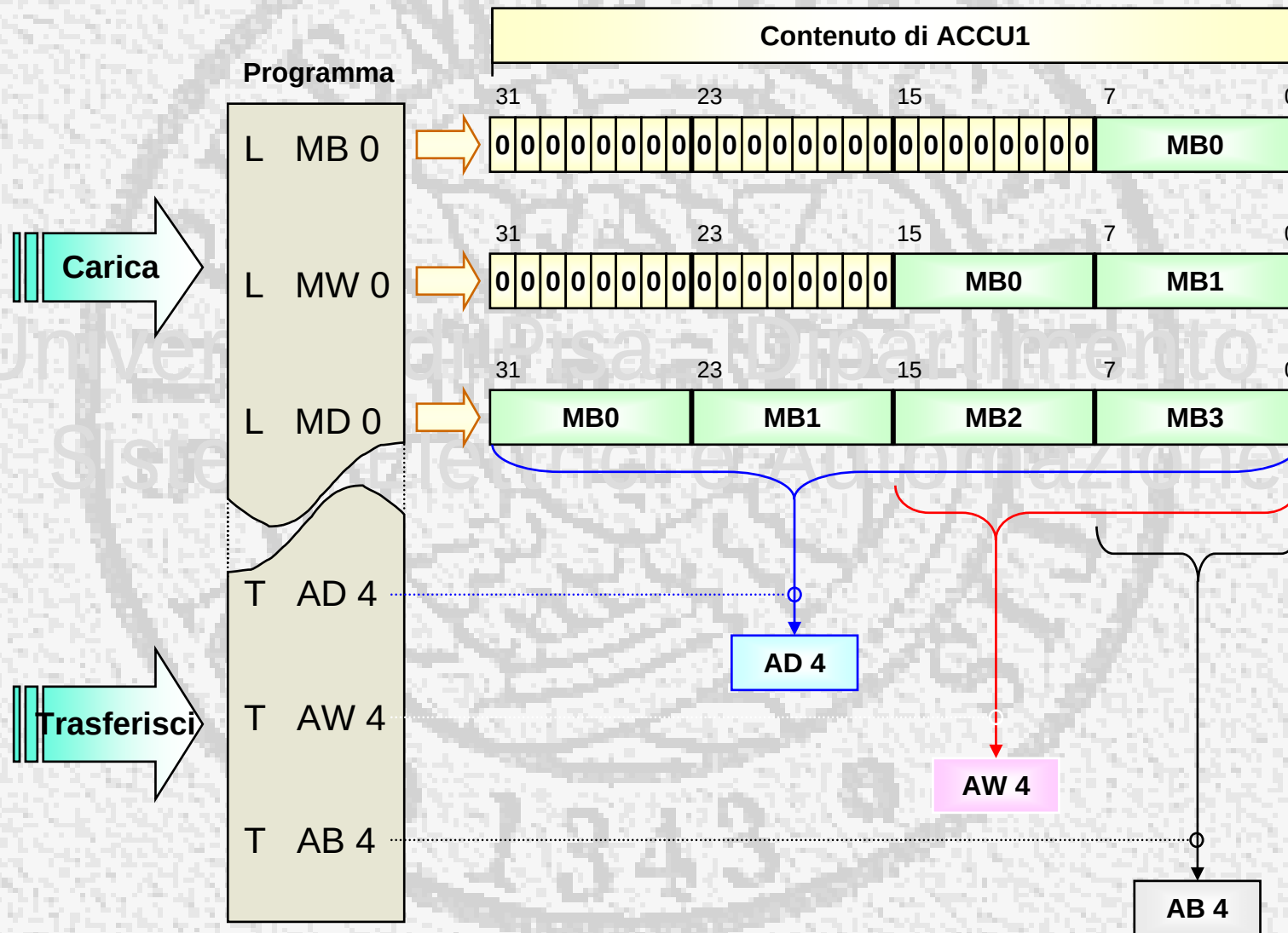


L +5	//	costante a16-bit (Integer)
L L#523123	//	costante a 32-bit (Double Integer)
L B#16#EF	//	byte in forma esadecimale
L 2#0010 0110 1110 0011	//	valore binario a16-bit
L 3.14	//	costante a 32-bit (Real)

Caricamento e Trasferimento Dati (2)

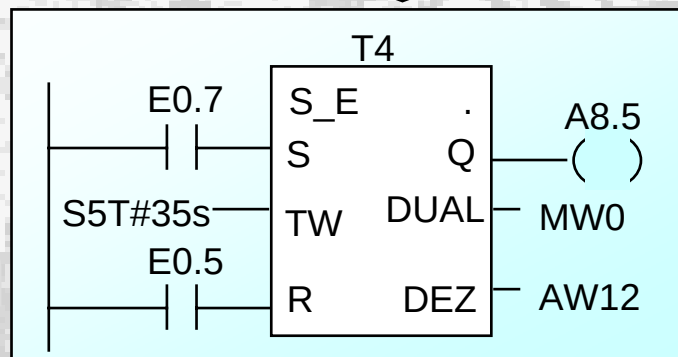


Caricamento e Trasferimento Dati



Temporizzatori: Ritardato all'Eccitazione (SE)

Ladder



AWL

```

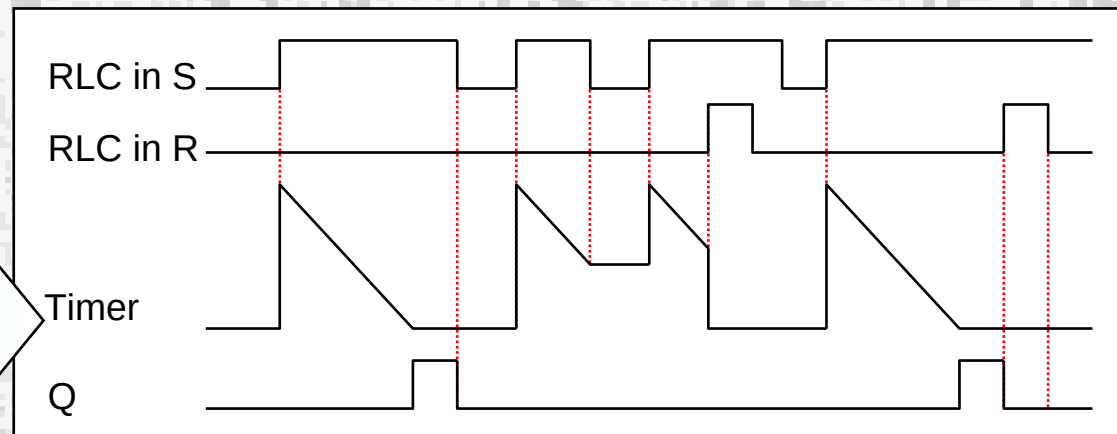
U  E0.7
L  S5T#35s
SE T4
U  E0.5
R  T4
L  T4
T  MW0
LC T4
T  AW12
U  T4
=  A8.5

```

Esempio

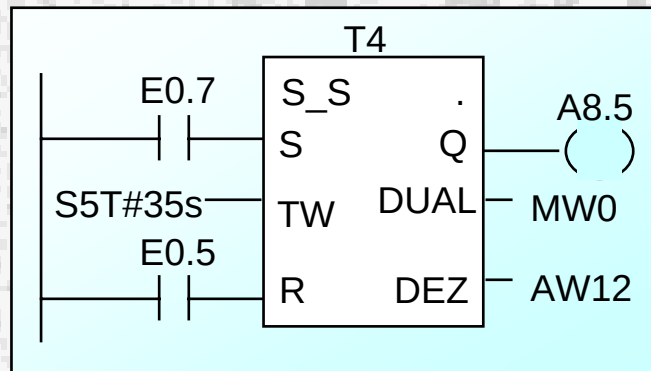
Timer

Q



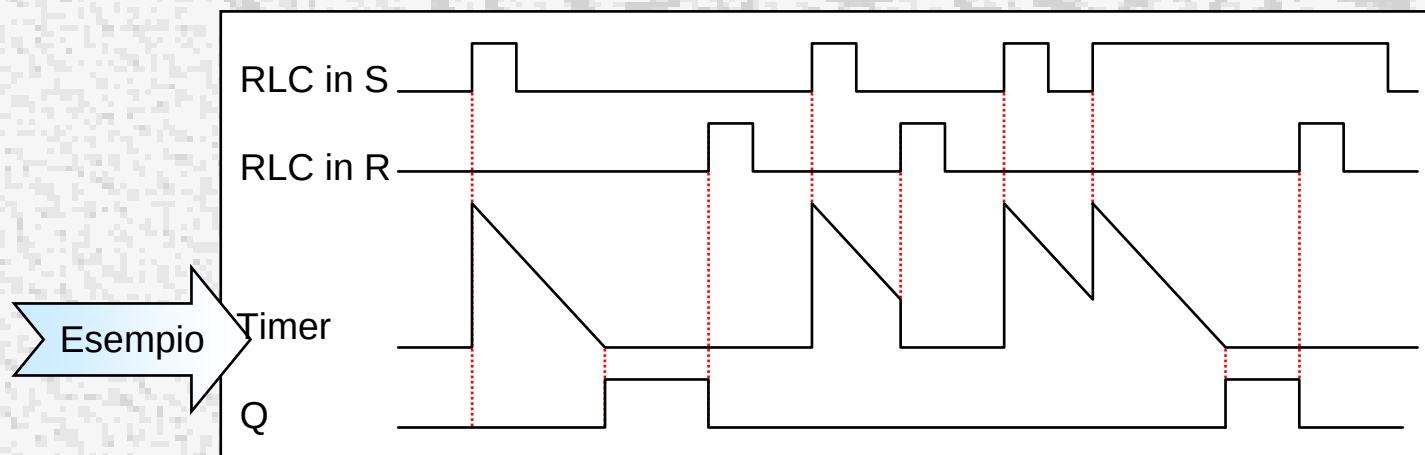
Temporizzatori: Ritardato all'Eccitazione con Memoria (SS)

Ladder



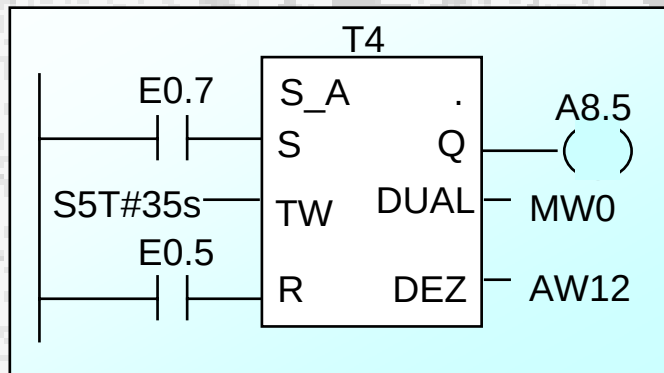
AWL

U	E0.7
L	S5T#35s
SS	T4
U	E0.5
R	T4
L	T4
T	MW0
LC	T4
T	AW12
U	T4
=	A8.5



Temporizzatori: Ritardato alla Diseccitazione (SA)

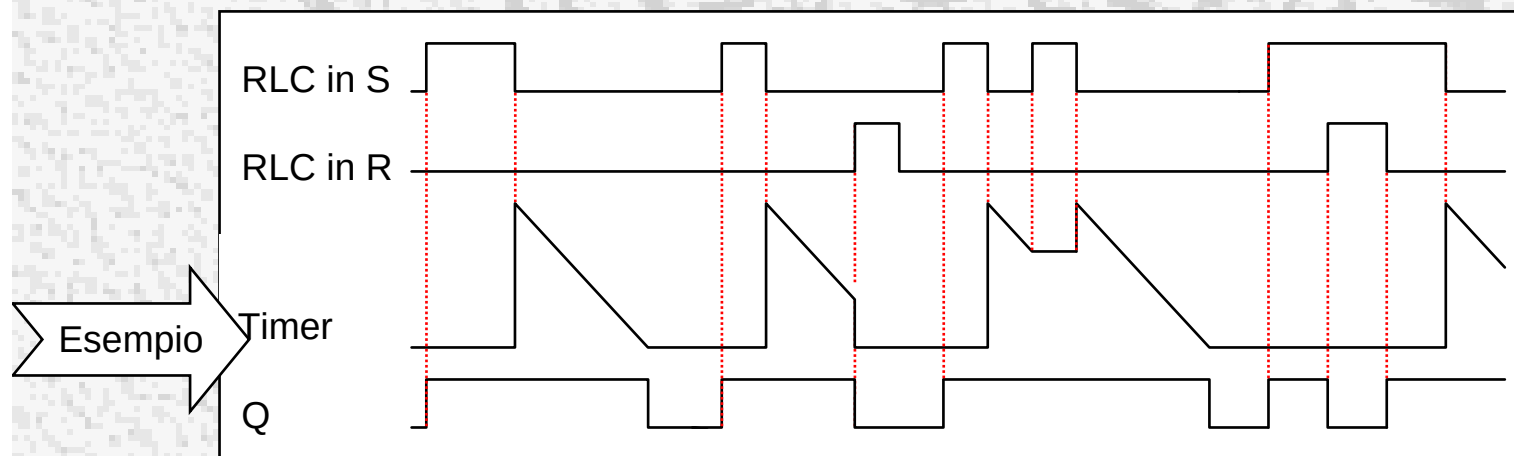
Ladder



AWL

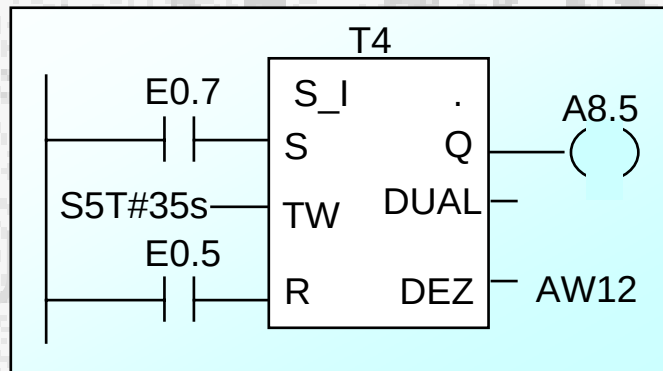
```

U   E0.7
L
S5T#35s
SA  T4
U   E0.5
R   T4
L   T4
T   MW0
LC  T4
T   AW12
U   T4
=   A8.5
  
```



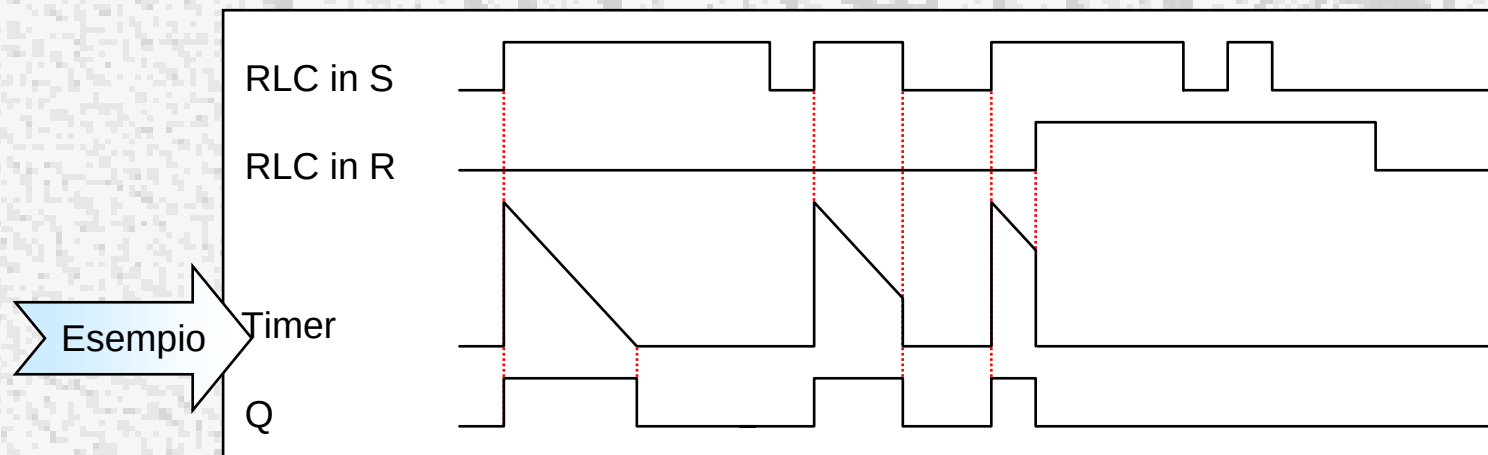
Temporizzatori: Impulsivo (SI)

Ladder



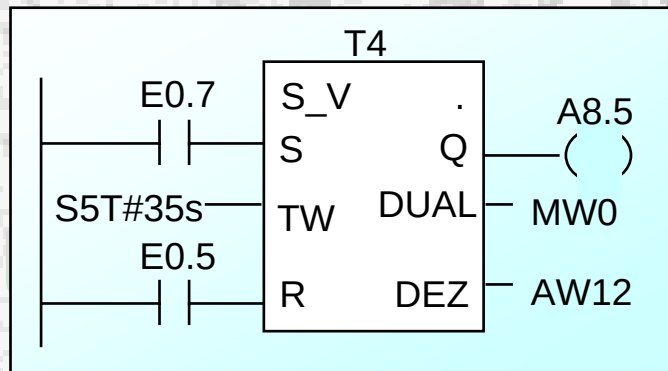
AWL

U	E0.7
L	S5T#35s
SI	T4
U	E0.5
R	T4
L	T4
T	MW0
LC	T4
T	AW12
U	T4
=	A8.5



Temporizzatori: ad Impulso Prolungato (SV)

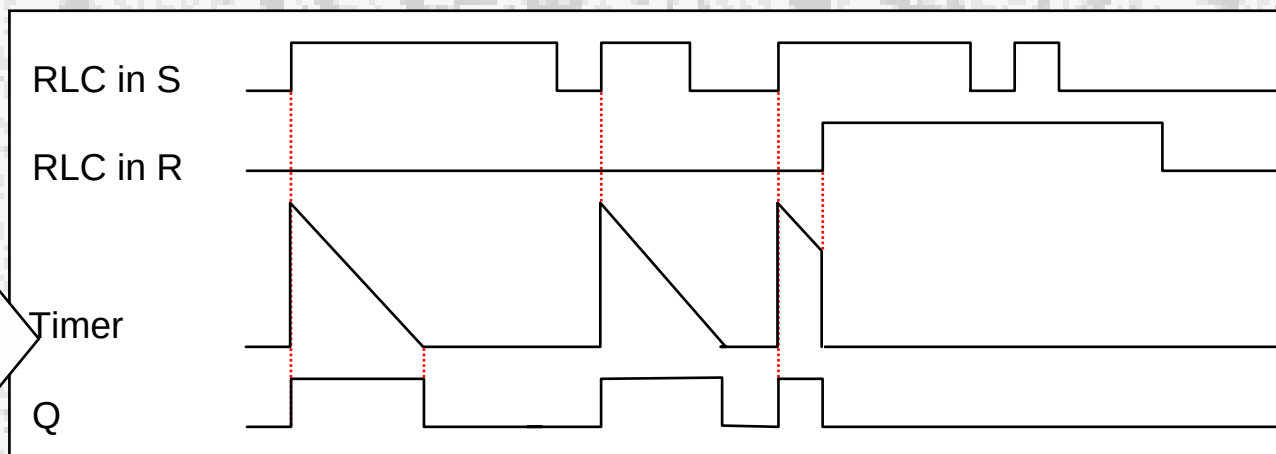
Ladder



AWL

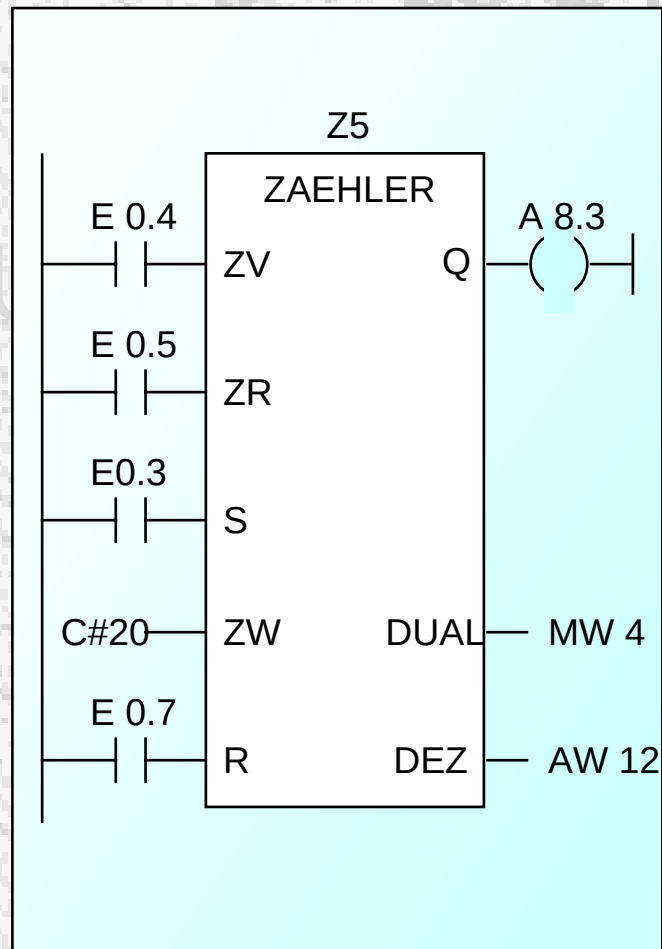
U	E0.7
L	S5T#35s
SV	T4
U	E0.5
R	T4
L	T4
T	MW0
LC	T4
T	AW12
U	T4
=	A8.5

Esempio



Contatori S5 in STEP 7

Ladder

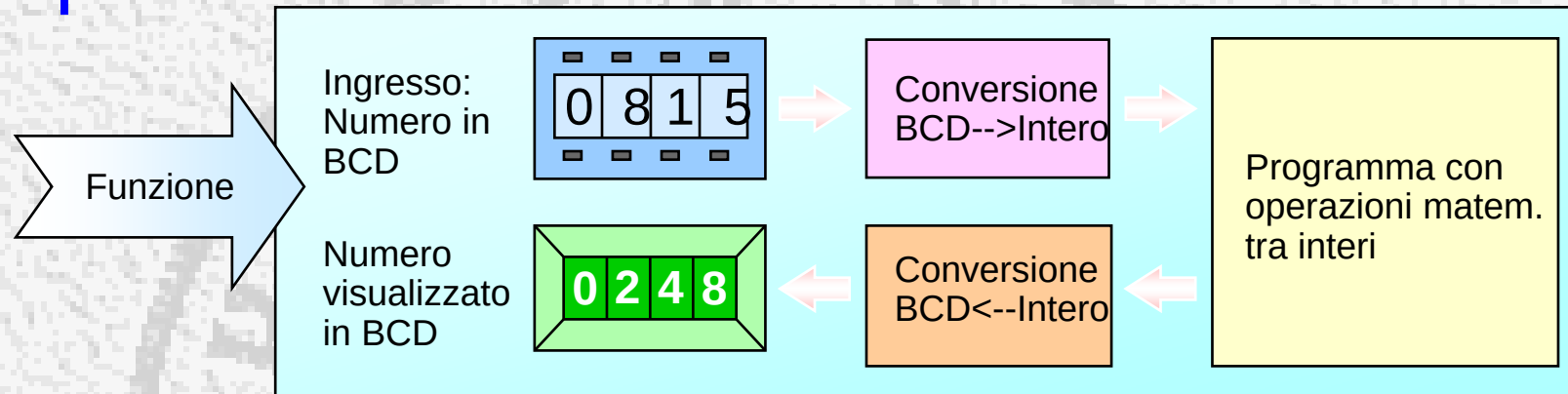


AWL

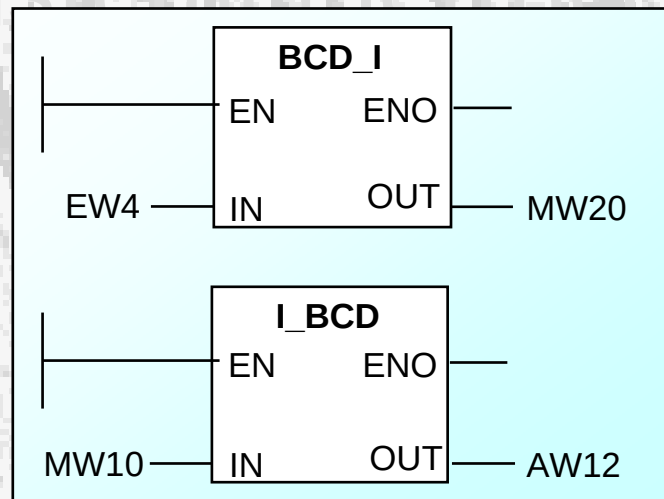
```

U   E0.4
ZV  Z5
U   E0.5
ZR  Z5
U   E0.3
L   C#20
S   Z5
U   E0.7
R   Z5
L   Z5
T   MW4
LC  Z5
T   AW12
U   Z5
=   A8.3
  
```

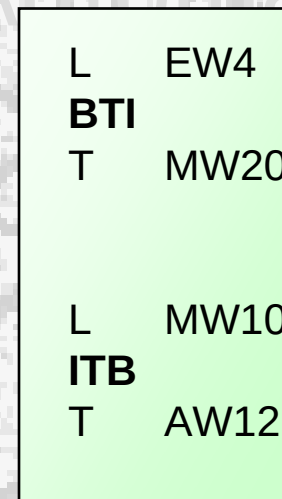
Operazioni di Conversione BCD <--> Intero



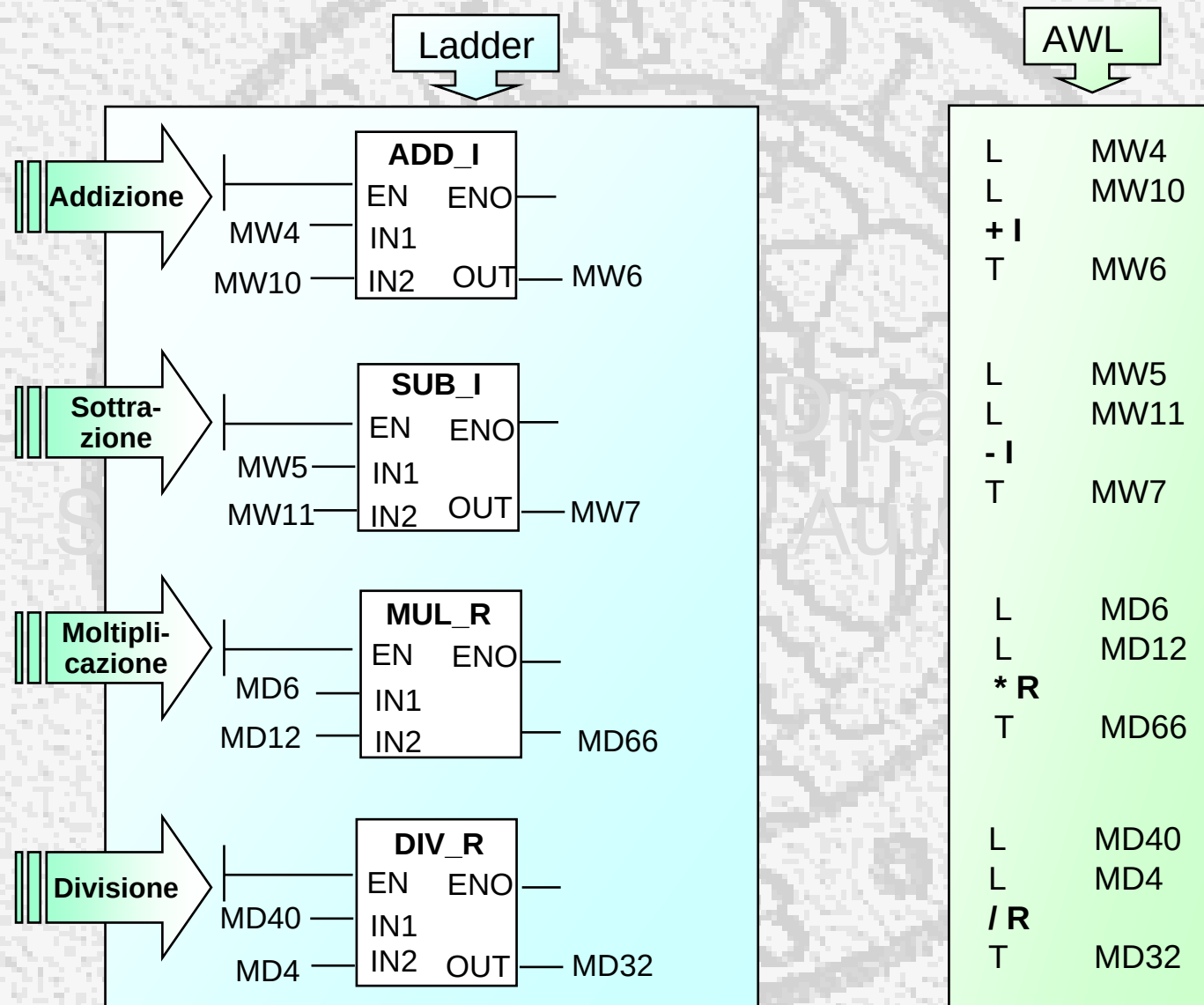
Ladder



AWL



Funzioni Matematiche di base



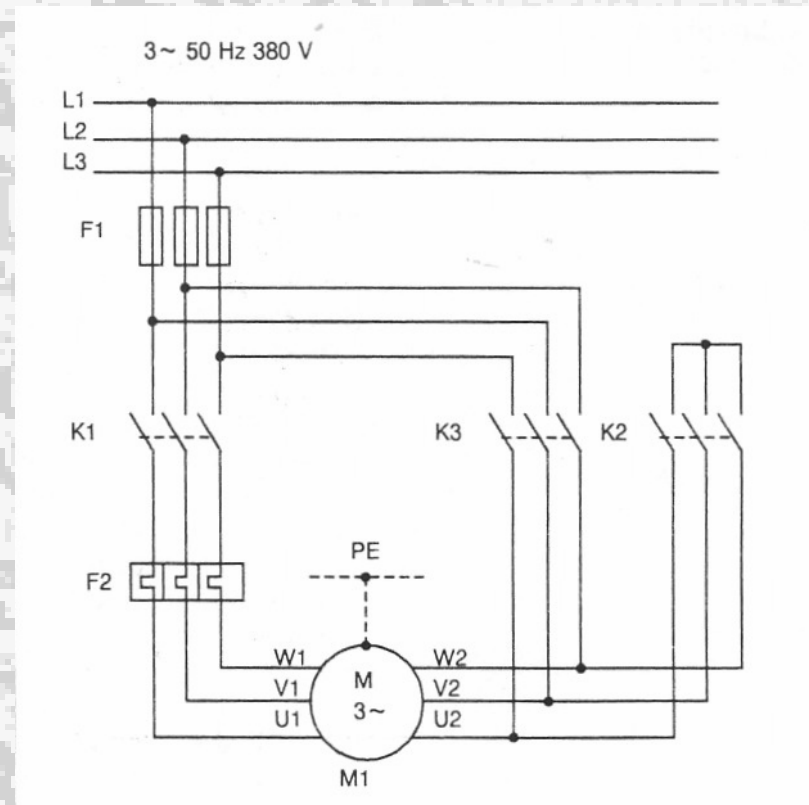
Esempio 3. Avviamento Stella-Triangolo

Descrizione del problema

Un motore asincrono viene inserito premendo il pulsante S1 (pulsante di marcia) e disinserito con il pulsante S0 (pulsante di arresto). Il motore si avvia con gli avvolgimenti statorici a stella. Dopo 10 s si ha commutazione degli avvolgimenti a triangolo.

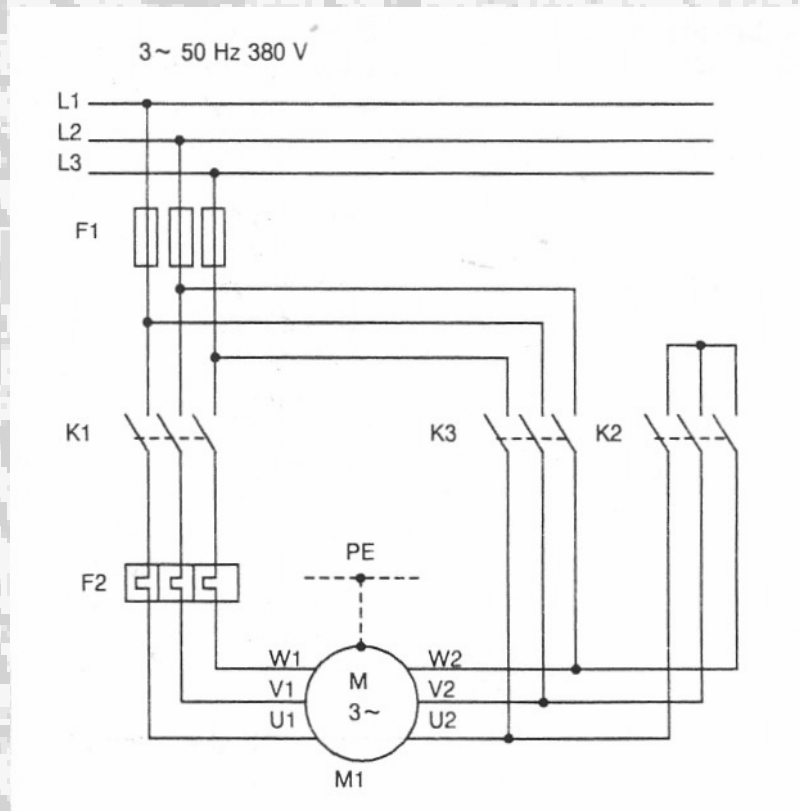
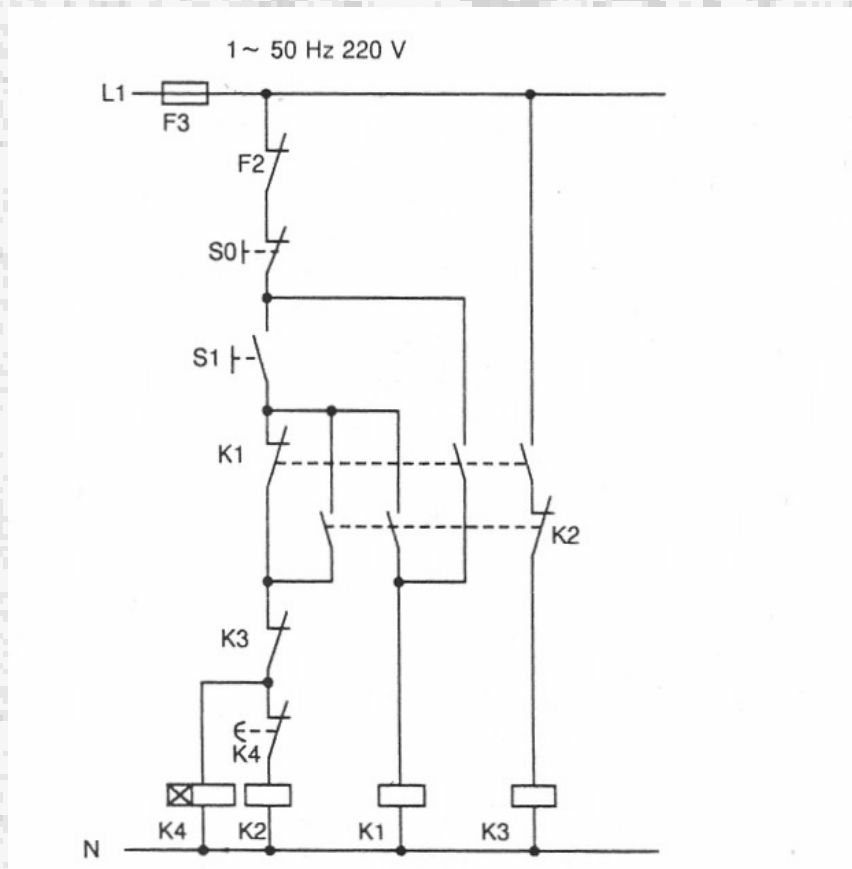
Le lampade di segnalazione H1, H2 e H3 visualizzano lo stato di attività del motore.

Il motore è protetto con un interruttore termico di sovracorrente e con fusibili per il corto circuito.



Esempio 3. Avviamento Stella-Triangolo

In logica cablata il circuito di controllo è:



K4 contatti temporizzatore

Esempio 3. Avviamento Stella-Triangolo

Si definiscono gli ingressi e si associano ai corrispondenti ingressi del PLC.

Simbolo	Variabile PLC	Descrizione
S0	E 0.1	OFF Pulsante N.A.
S1	E 0.2	ON Pulsante N.A.

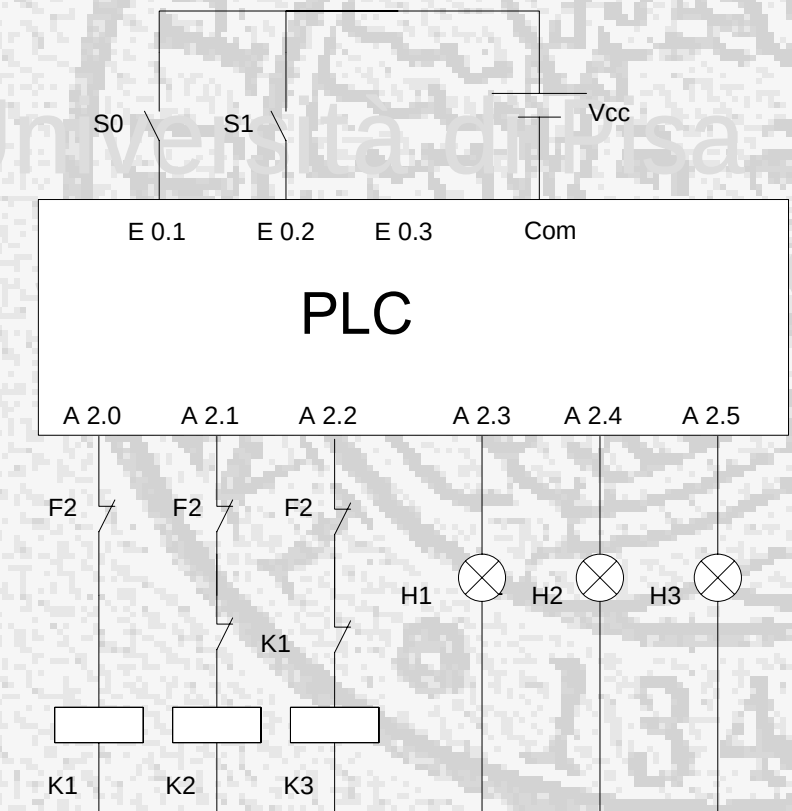
Esempio 3. Avviamento Stella-Triangolo

Si definiscono le uscite e si associano alle corrispondenti uscite del PLC.

Simbolo	Variabile PLC	Descrizione
K1	A 2.0	Bobina contattore generale
K2	A 2.1	Bobina contattore a stella
K3	A 2.2	Bobina contattore a triangolo
H1	A 2.3	Segnalazione motore fermo
H2	A 2.4	Segnalazione collegamento a stella
H3	A 2.5	Segnalazione collegamento a triangolo

Esempio 3. Avviamento Stella-Triangolo

Lo schema di collegamento al PLC è il seguente:

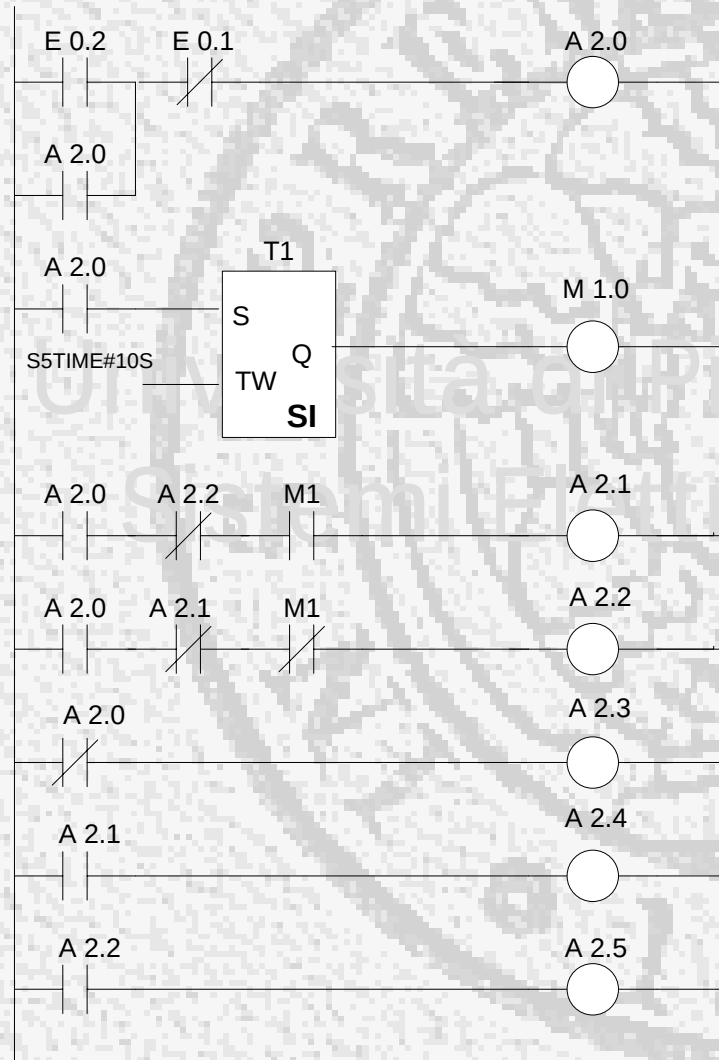


I contatti dell'interruttore termico devono essere collegati sulle linee di alimentazione dei tre contattori.

Inoltre un contattore di commutazione non deve poter essere inserito fintanto che l'altro non è completamente disinserito. La concomitanza di entrambi i comandi di eccitazione potrebbe verificarsi a causa delle velocissime reazioni di comando elettronico rispetto alle reazioni delle parti meccaniche in movimento.

Esempio 3. Avviamento Stella-Triangolo

Il programma ladder è:



Avviamento/arresto del motore

Temporizzatore attivo per 10 s

Contattore stella

Contattore triangolo

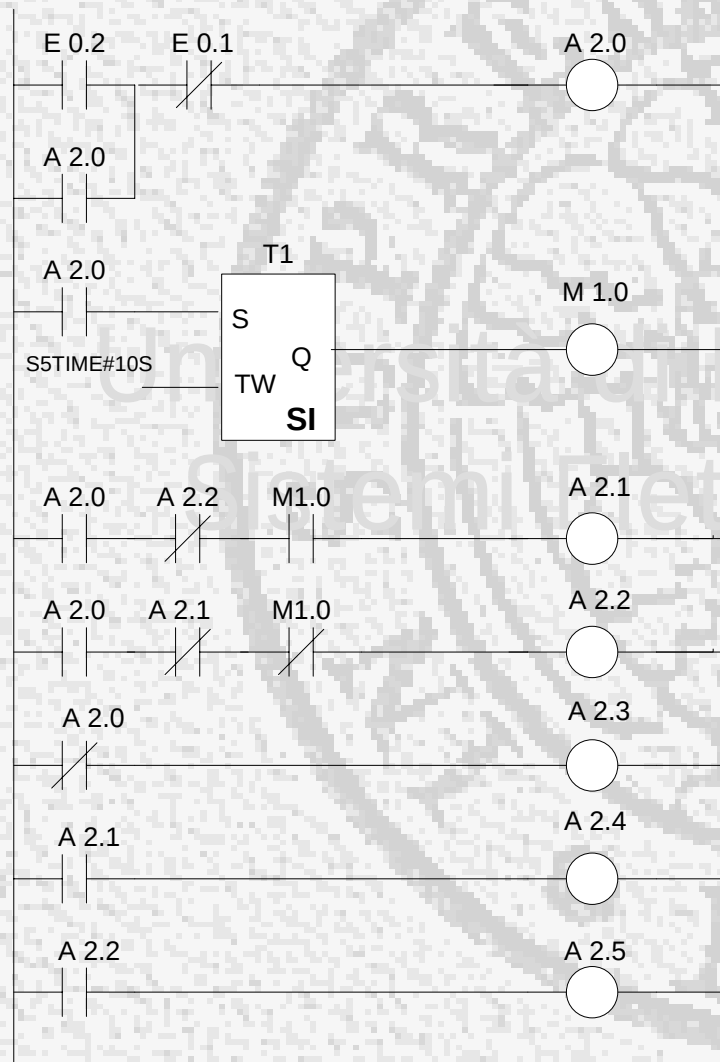
Luce impianto spento

Luce collegamento a stella

Luce collegamento a triangolo

Esempio 1. Avviamento/Arresto di un motore

Funzioni logiche



$$(E\ 0.2 + A\ 2.0) * \overline{E\ 0.1} = A\ 2.0$$

$$A\ 2.0 * \overline{A\ 2.2} * M1 = A\ 2.1$$

$$A\ 2.0 * \overline{A\ 2.1} * \overline{M1} = A\ 2.2$$

$$\overline{A\ 2.0} = A\ 2.3$$

$$A\ 2.1 = A\ 2.4$$

$$A\ 2.2 = A\ 2.5$$

Il codice AWL è:

U(	SI T1	UN A 2.1	= A 2.5
O E 0.2	U T1	UN M 1.0	
O A 2.0	= M 1.0	= A 2.2	
)	U A 2.0	UN A 2.0	
UN E 0.1	UN A 2.2	= A 2.3	
= A 2.0	U M1	U A 2.1	
U A 2.0	= A 2.1	= A 2.4	
L S5T#10S	U A 2.0	U A 2.2	

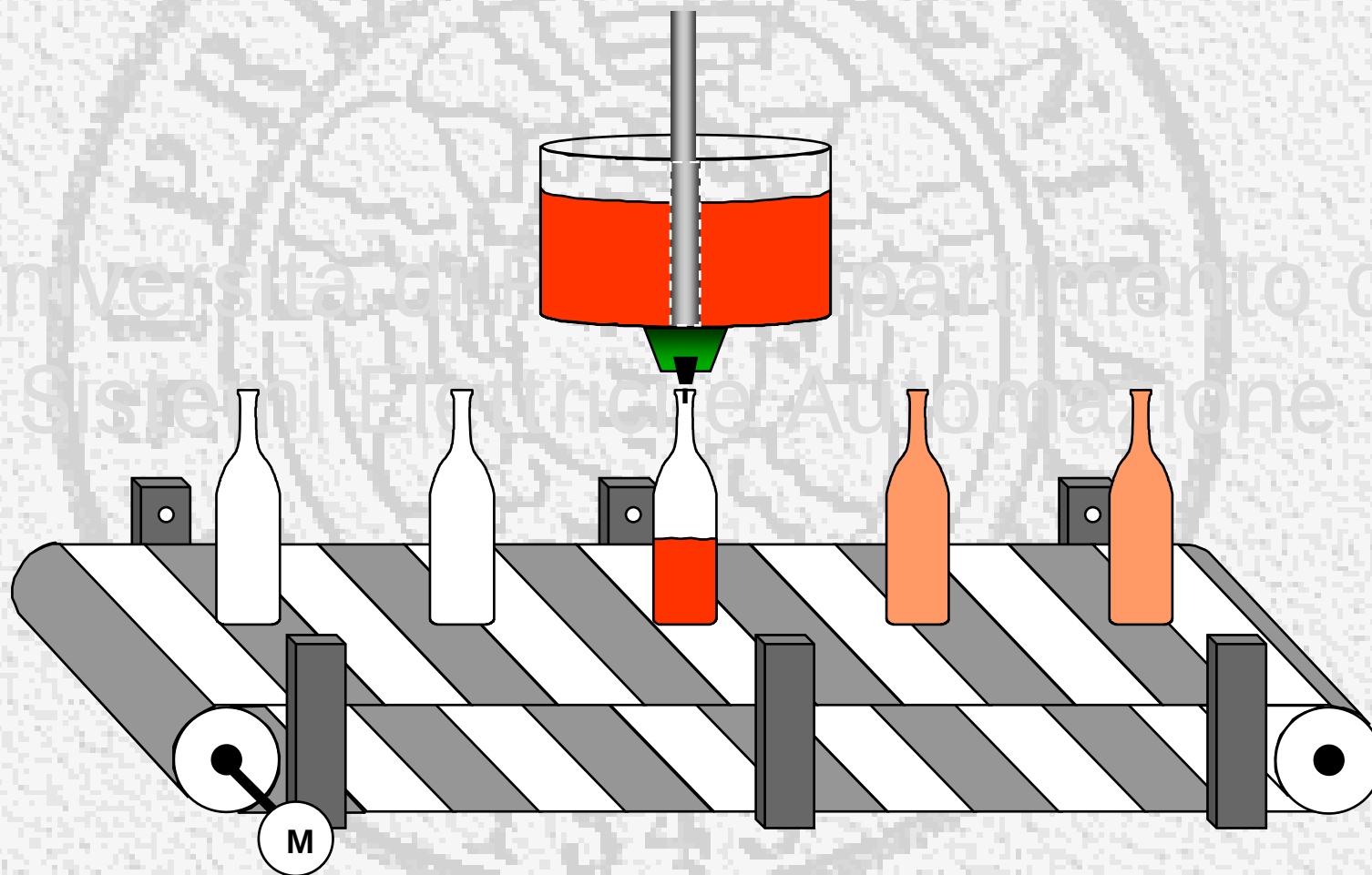
Applicazione

- Gestione di un impianto di imbottigliamento

Schema ladder

Linguaggio testuale

Esercizio: Gestione di un impianto d'imbottigliamento



Descrizione dell'impianto

L'impianto deve essere acceso con un pulsante di Start e arrestato con un pulsante di Stop

Sono previste due modalità di funzionamento: Manuale e Automatico, selezionabili con un selettore.

In entrambe le modalità deve essere possibile comandare il movimento Avanti o Indietro del nastro trasportatore

Step n° 1: individuazione delle condizioni di funzionamento

Il nastro trasportatore si muove in avanti/indietro in modalità MANUALE se:

- a) l'impianto è acceso
- b) è selezionata la modalità manuale
- c) è dato il comando Avanti / Indietro

Il nastro trasportatore si muove in avanti/indietro in modalità AUTOMATICA se:

- a) l'impianto è acceso
- b) è selezionata la modalità automatica
- c) è dato il comando Avanti / Indietro

Step n° 1: individuazione delle condizioni di funzionamento

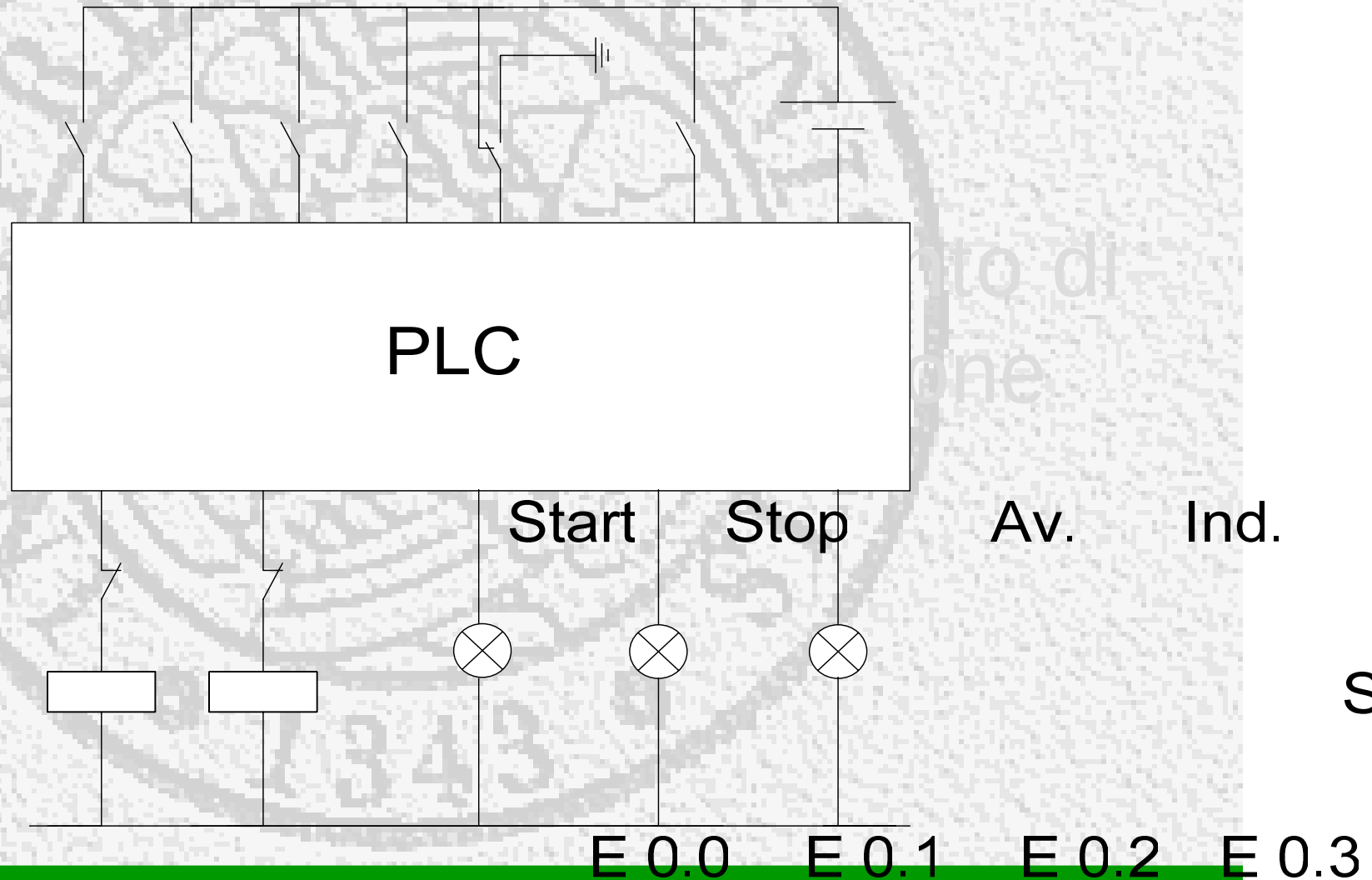
La selezione MANUALE / AUTOMATICA è realizzata con un commutatore a due stati e da un pulsante di conferma per ogni modalità

Università di Pisa - Dipartimento di
Sistemi Elettrici e Automazione

Step n° 2: definizione degli ingressi e delle uscite

Ingressi	Uscite
<u>Impianto ON/OFF</u>	A 4.1 Led impianto acceso
E 0.0 = Start (Pulsante NA)	A 4.2 Led Modalità manuale
E 0.1 = Stop (Pulsante NA)	A 4.3 Led Modalità automatica
<u>Modo Manuale/Automatico</u>	A 8.5 Nastro trasportatore avanti
E 0.4 = Manuale/Automatico (Selettore)	A 8.6 Nastro trasportatore indietro
E 0.5 = Comando di selezione (puls. NA)	
E 0.2 = Comando avanti (puls. NA)	
E 0.3 = Comando indietro (puls. NA)	

Step n° 3: definizione dei collegamenti al PLC



Step n° 4: definizione degli stati dell'impianto

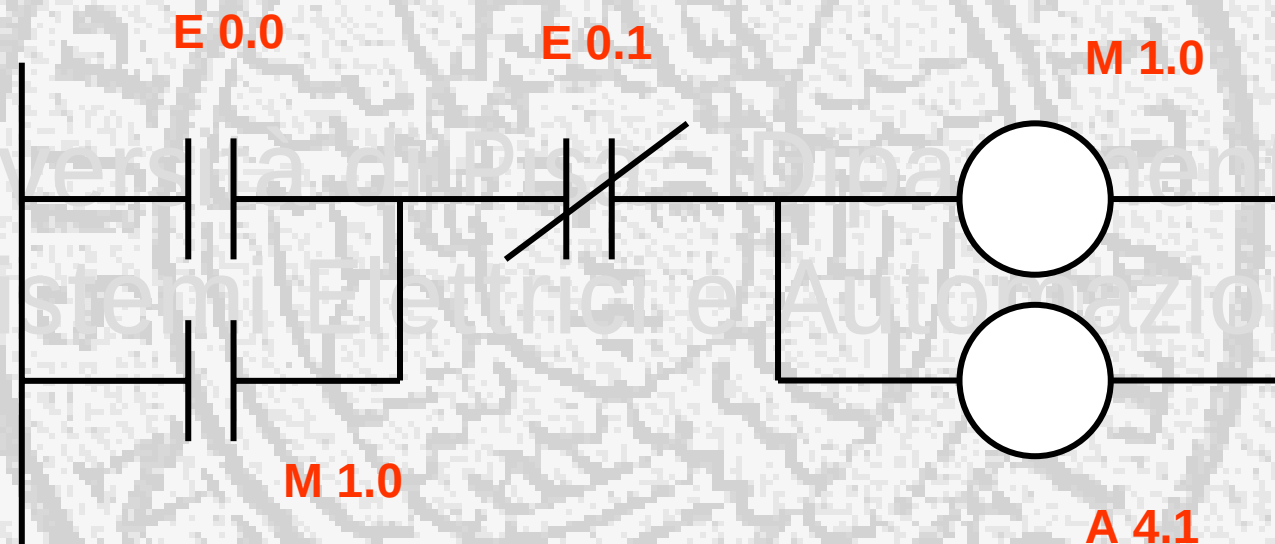
Si definiscono le configurazioni in cui l'impianto si viene a trovare:

- Impianto spento
- Impianto acceso

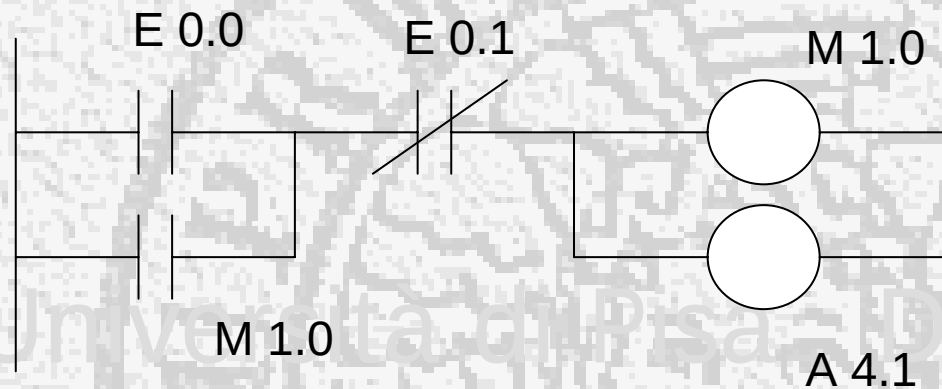
Modalità manuale – Modalità automatica

Movimento in avanti – Movimento indietro

Accensione / arresto dell'impianto



Accensione / arresto dell'impianto



Funzione Logica

$$(E0.0 + M1.0) * \underline{E0.1} = M1.0$$

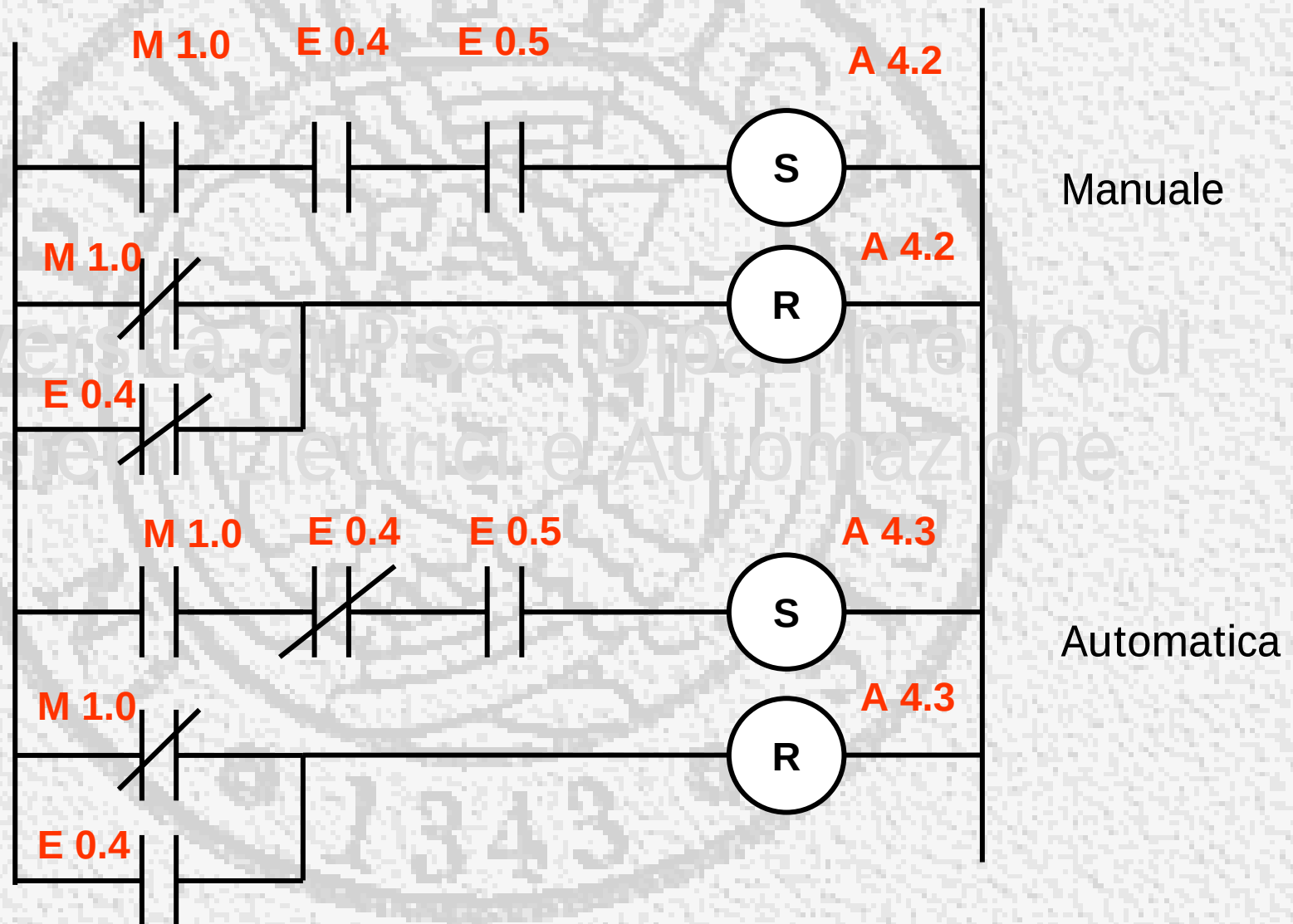
$$(E0.0 + M1.0) * \underline{E0.1} = A4.1$$

Codice AWL

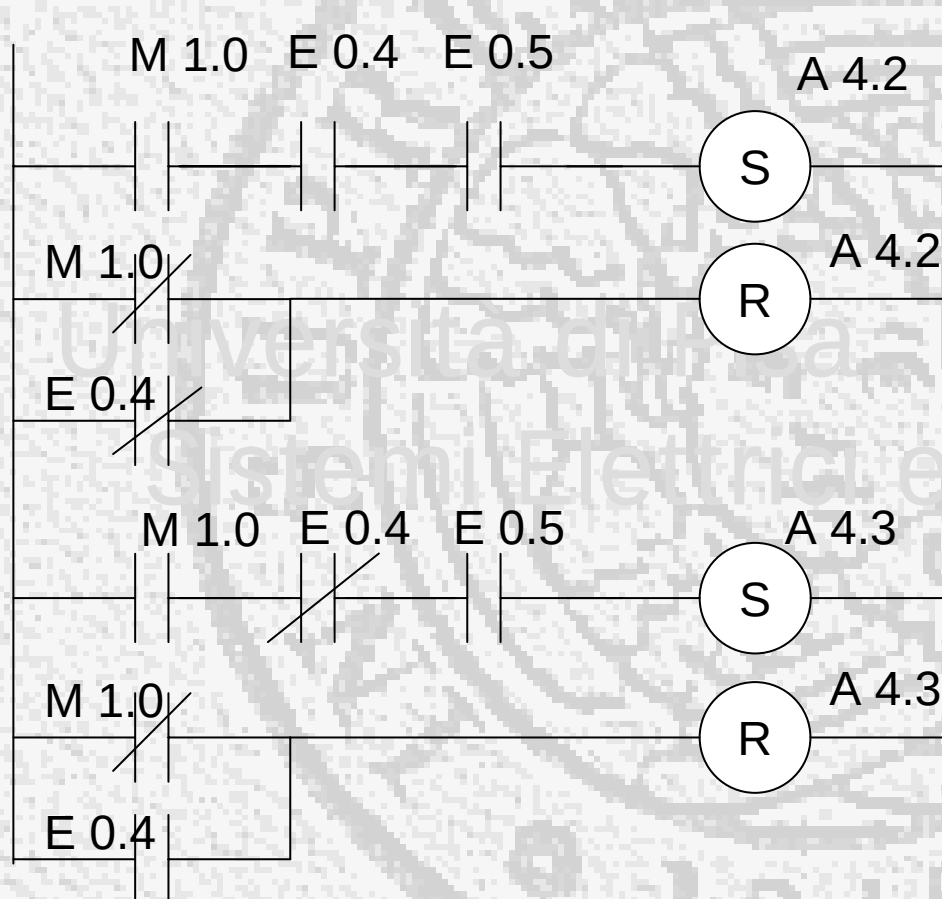
```

U(
O E0.0
O M1.0
)
UN E0.1
=M1.0
=A4.1
  
```

Selezione modalità Manuale / Automatica



Selezione modalità Manuale / Automatica



Funzione Logica

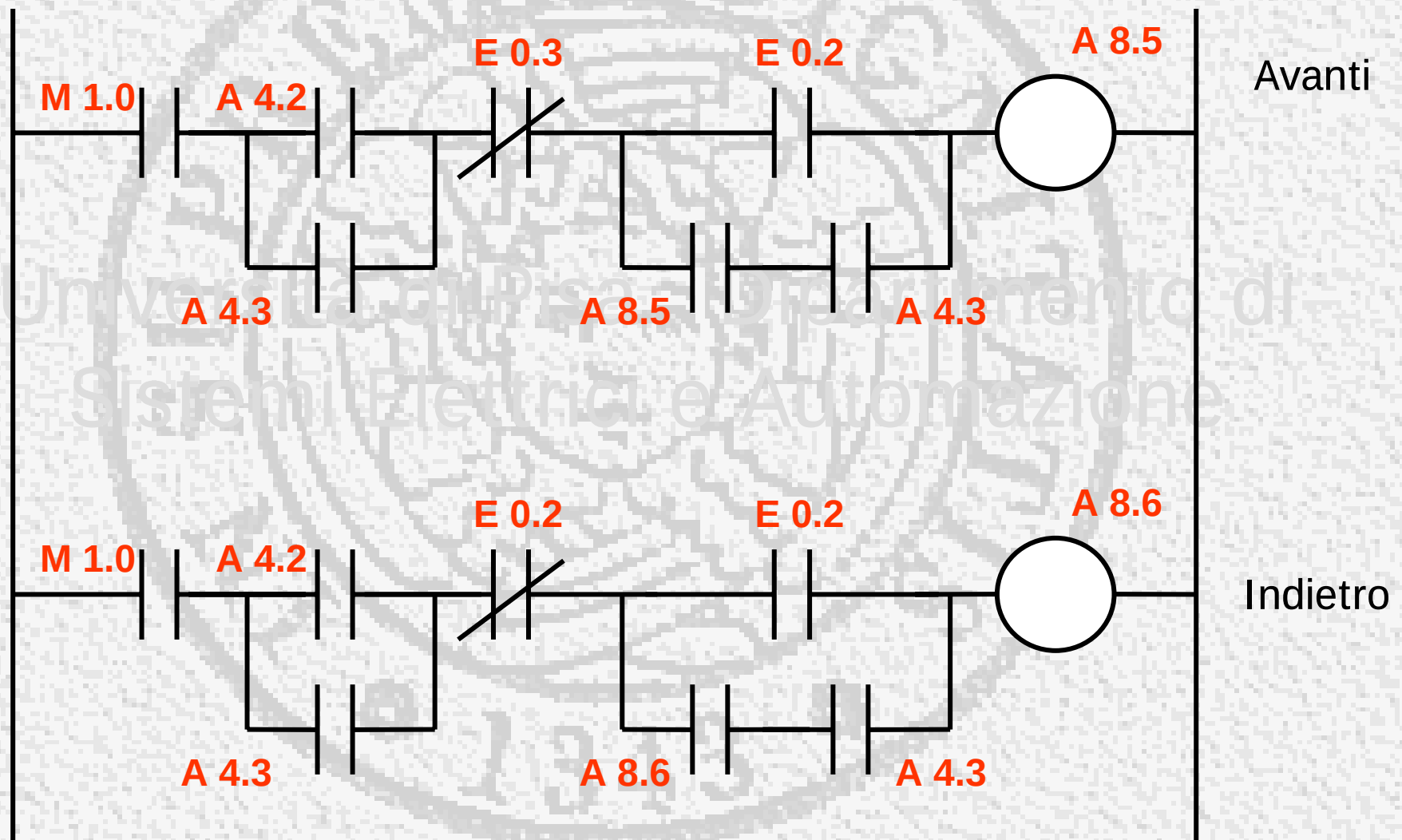
$M1.0 * E0.4 * E0.5 = S(A\ 4.2)$
 $(\underline{M1.0} + \underline{E0.4}) = R(A\ 4.2)$

$M1.0 * \underline{E0.4} * E0.5 = S(A\ 4.3)$
 $(\underline{M1.0} + E0.4) = R(A\ 4.3)$

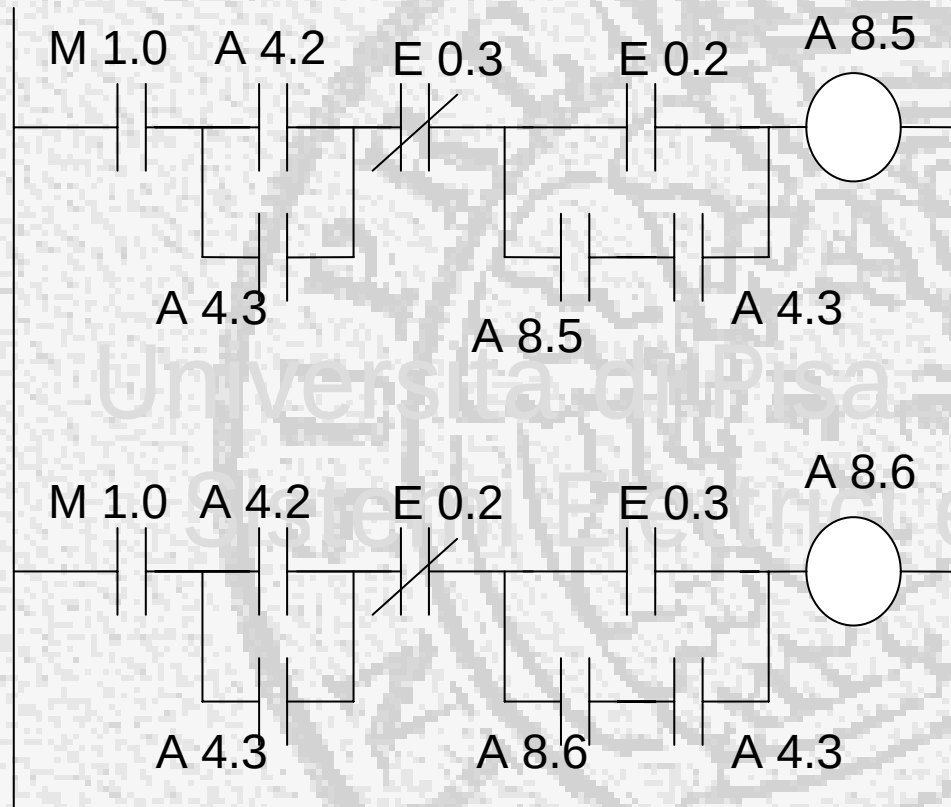
Codice AWL

U M1.0	U M1.0
U E0.4	UN E0.4
U E0.5	U E0.5
S A4.2	S A4.3
ON M1.0	ON M1.0
ON E0.4	O E0.4
R A4.2	R A4.3

Movimento Avanti / Indietro



Movimento Avanti / Indietro



Funzione Logica

$$M1.0 * (A4.2 + A4.3) * \overline{E0.3} * (E0.2 + (A8.5 * A4.3)) = A8.5$$

$$M1.0 * (A4.2 + A4.3) * \overline{E0.2} * (E0.3 + (A8.6 * A4.3)) = A8.6$$

U M1.0	O E0.2
U(	O(
O A4.2	U A8.5
O A4.3	U A4.3
)	))
UN E0.3	= A8.5
U(	