

Realizzazione della Macchina Astratta

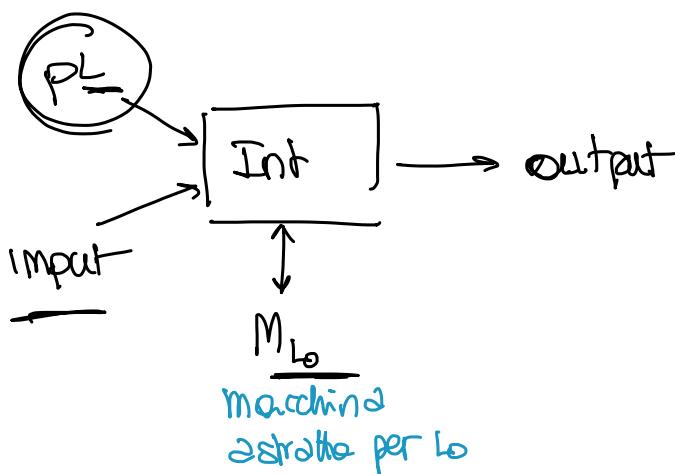
Soluzione Interpretativa

Soluzione Compilativa

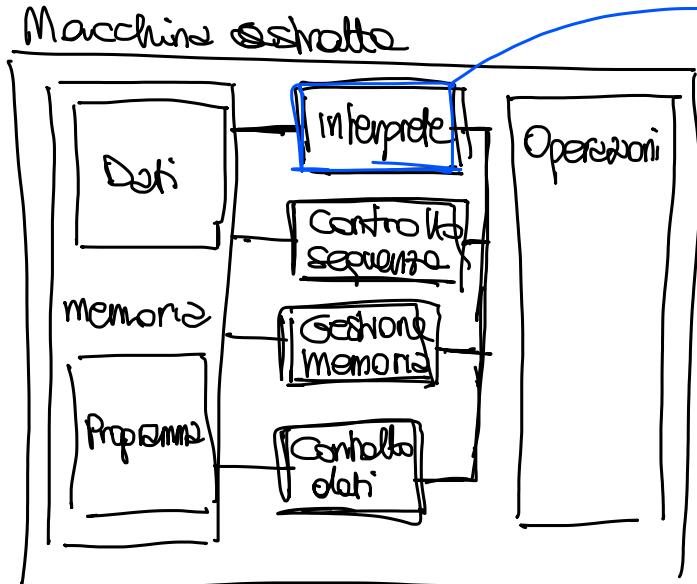
Interprete da L o Lo

$$I^{Lo}(P^L, \text{input}) = \text{output}$$

$$I^{Lo} : \text{Prog}^L \times D \rightarrow D$$



Macchina astratta



begin

go:=true;

while go do begin

FETCH (OPCODE, OPINFO) at PC
DECODE (OPCODE, OPINFO)

if OPCODE needs ARGS
then FETCH (ARGS)

case OPCODE of

OP1 : EXECUTE (OP1, ARGS);

OP2 : EXECUTE (OP2, ARGS);

:

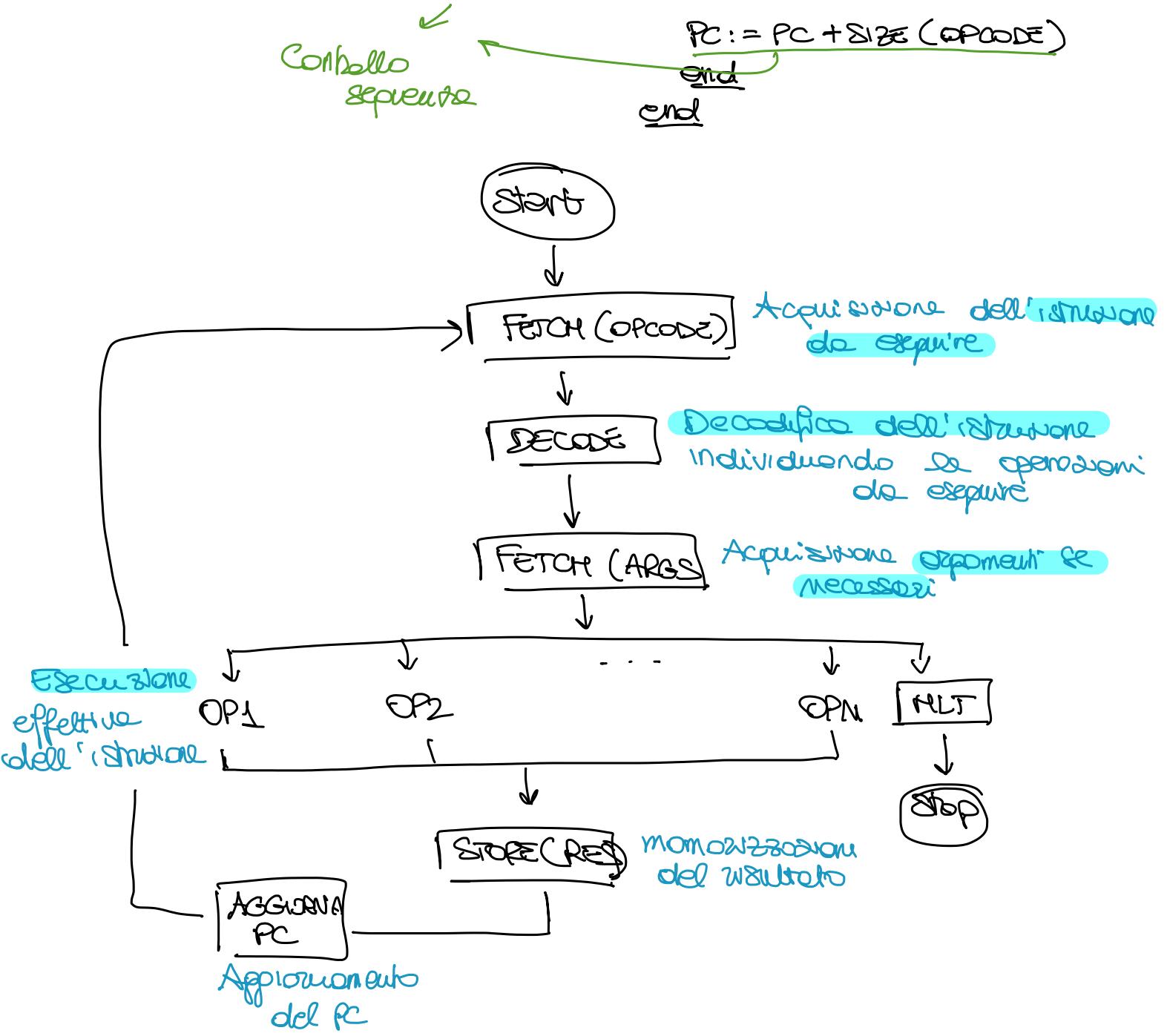
OPM : EXECUTE (OPM, ARGS);

HLT : go := false;

If OPCODE has result then

STORE (RES)

controllo dati



SOLUZIONE COMPILATIVA

→ Macchina Astratta = compilatore

COMPILATORE : Un compilatore da L o Lo ling. sorgente
 \downarrow ling. target

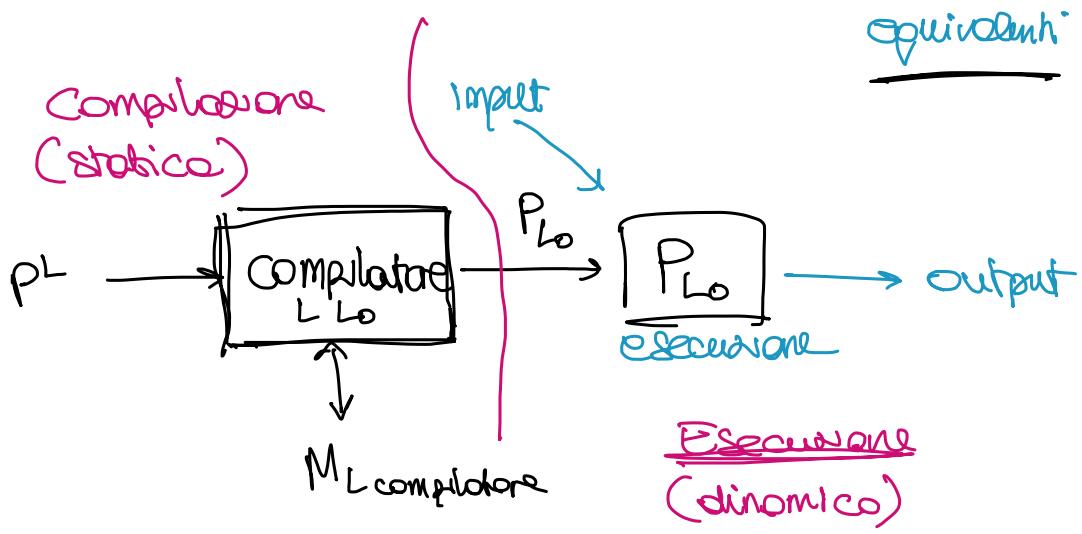
TRADUTTORE è un programma $C^{Lo} : \text{Prog}^L \rightarrow \text{Prog}^{Lo}$

$$C^{Lo}(P^L) = P^{Lo} \text{ tale che } \forall d \in D$$

$$P^L(d) = P^{Lo}(d)$$

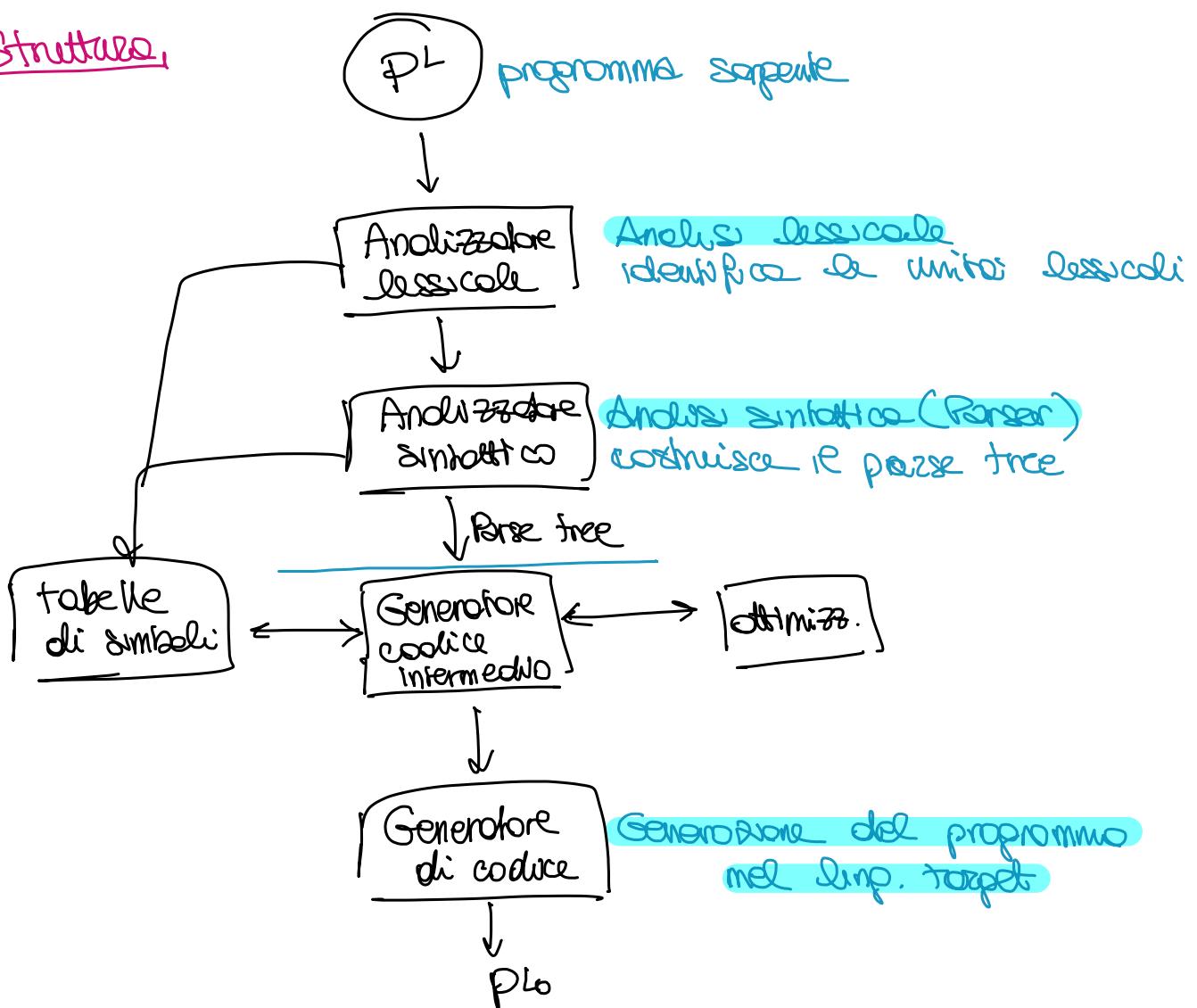
Equivalente

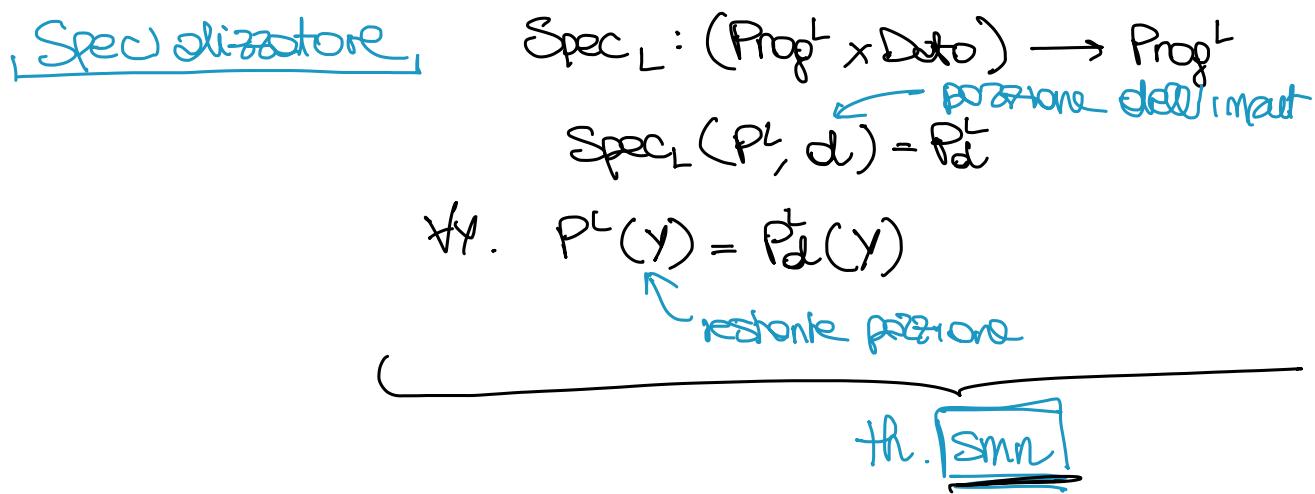
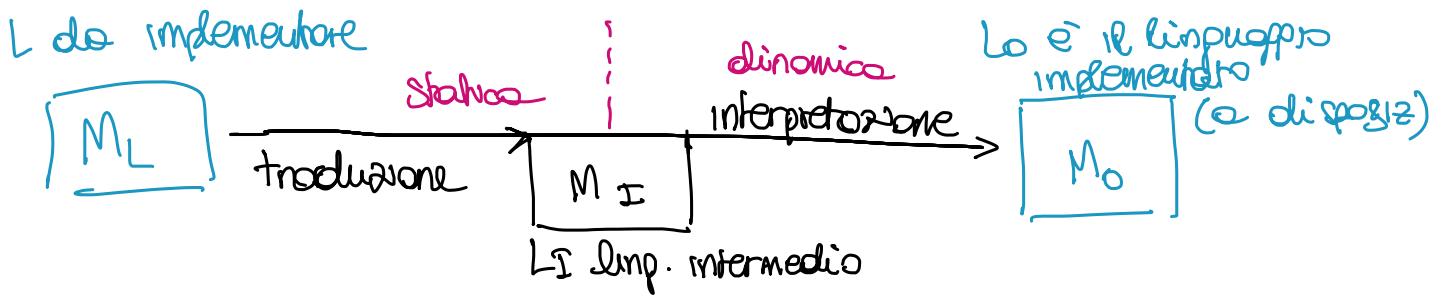
P^L e P^{Lo} sono semanticamente



Macchina assesta
del linguaggio
usato per scrivere
il compilatore

Struttura





Speciлязатор un interprete su un programma

→ un compilatore
otteniamo

DESCRIVERE I PL

→ **SINTASSI** → Regole di costruzione di frasi ben formate

⇒ descrive relazioni tra simboli

→ **SEMANTICA** → Attribuzione del significato ai simboli

⇒ Date una frase che rispetta le regole sintattiche gli attribuisce un significato

→ **PRAGMATICA** → Regole (non obbligatorie) di buon uso (ingegneria del SW)

→ **IMPLEMENTAZIONE** → Come si esprimono frasi ben formate che hanno significato

Sintassi → descrivere le regole di formazione

Parola stringe caratteri

Frase sequenza di parole ben formate

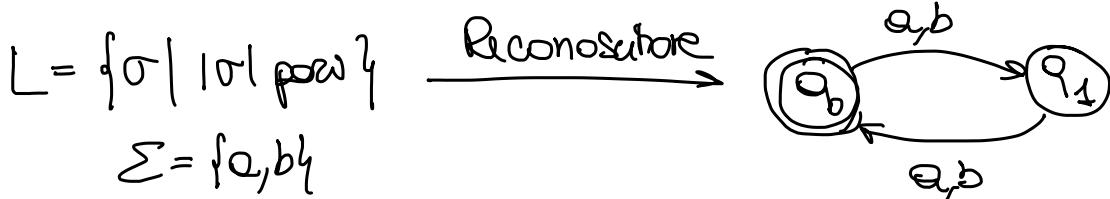
Linguaggio insieme delle frasi

Lessema parola con significato specifico (simboli terminali della grammatica) unità sintattica del linguaggio

Token categoria sintattica (simboli non terminali)

Riconoscitori
(Automi)

Generato da
(Grammatiche)



$$G: S \rightarrow \varepsilon \mid aaS \mid abS \mid baS \mid bbS$$

Descrizione della grammatica

→ Alfabeto (simboli)

→ Descrizione lessicale (sequenze di simboli che costruiscono parole del linguaggio
→ "simboli" terminali)

→ Regole sintattiche (Regole di derivazione della grammatica che compongono tra loro parole)

→ Categorie sintattiche ("simboli" non terminali della grammatica)

⇒ Grammatiche CFG (BNF)

$$\text{CFG} \quad G = \langle V, T, P, S \rangle$$

V insieme finito di simboli non terminali
elementi della frase (token)
categorie sintattiche → Espressioni
... → Comandi
... → D dichiarazioni
... → Procedure

T insieme finito di simboli terminali

vocabolario del linguaggio (esempi)
while, if, := - {, }, + - ...

P produzioni $A \rightarrow \alpha \quad A \in V \quad \alpha \in (V \cup T)^*$

Regole di formazione del programma

Com → ... while Exp do Com ...

Exp ...

SE V simbolo iniziale Categorie dei programmi

Esempio

Exp

$\langle \{E\}, \{\text{or, and, not, (,), 0, 1}\}, P \rangle$

$E \rightarrow 0 | 1 | (\underline{E} \text{ or } \underline{E}) |$
 $(\underline{E} \text{ and } \underline{E}) | (\text{not } E)$