Schema di traduzione on-the-fly: Translator

NON TERMINALE		PRODUZIONE
<prog>	→	{prog.next = newlabel(), statlist.next = prog.next} **<statlist>** {emitlabel(prog.next)} EOF
<statlist>	→	{stat.next = newlabel()} **<stat>** {statlistp.next = statlist.next} **<statlistp>**
<statlistp>	→	; {stat.next = newlabel()} **<stat>** {statlistp1.next = statlistp.next} **<statlistp1>**
<statlistp>	→	ε
<stat>	→	ID := **<expr>** {emit(istore,ID)}
<stat>	→	print (**<expr>**) {print()}
<stat>	→	read (ID) {read(ID)}
<stat>	→	case {whenlist.next = newLabel(),whenlist.end = stat.next()} **<whenlist>** else {stat1.next = stat.next} **<stat1>**{emitLabel(stat1.next)}
<stat>	→	while ({bexpr.true = fall, bexpr.false = stat.next, stat1.next = newlabel(), emitLabel(stat1.next)} **<bexpr>**) **<stat1>** {emit(goto stat1.next), emitLabel(stat.next)}
<stat>	→	{ {statlist.next = stat.next} **<statlist>** }
<whenlist>	→	{whenitem.next = newLabel()} **<whenitem>** {emit(goto,whenlist.end), emitLabel(whenitem.next), whenlistp.end = whenlist.end, whenlistp.next = whenlist.next } **<whenlistp>**
<whenlistp>	→	{whenitem.next = newLabel()} **<whenitem>** {emit(goto,whenlistp.end), emitLabel(whenitem.next), whenlistp1.end = whenlistp.end, whenlistp1.next = whelistp.next} **<whenlistp1>**

<whenlistp>	→	ε
<whenitem>	→	when ({bexpr.true = fall, bexpr.false = whenitem.next} **<bexpr>**) {stat.next = whenitem.next} **<stat>**
<bexpr>	→	**<expr>** RELOP **<expr>** {if(bexpr.true = fall) then emit(if_icmpNOTrel, bexpr.false) else if(bexpr.false = fall) then emit(if_icmprel, bexpr.true) else emit(if_icmprel,bexpr.true) emit(goto,bexpr.false)}
<expr>	→	**<term> <exprp>**
<exprp>	→	+ **<term>** {emit(iadd)} **<exprp>**
<exprp>	→	- **<term>** {emit(isub)} **<exprp>**
<exprp>	→	ε
<term>	→	**<fact> <termp>**
<termp>	→	* **<fact>** {emit(imul)} **<termp>**
<termp>	→	/ **<fact>** {emit(idiv)} **<termp>**
<termp>	→	ε
<fact>	→	(**<expr>**)
<fact>	→	NUM {emit(NUM.val)}
<fact>	→	ID {emit(ID.addr)}