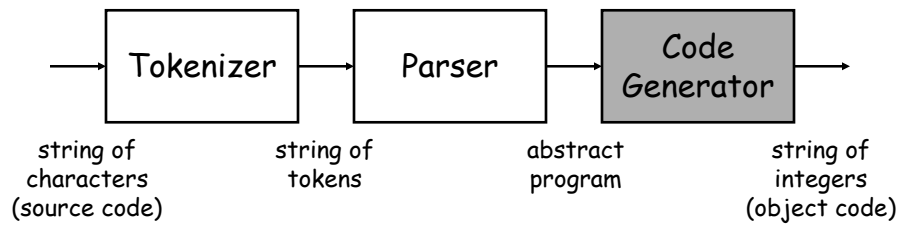


## *Translator Architecture*



## *Execute\_Program*

```
procedure_body Execute_Program (  
    preserves Program& p  
)  
{  
  
}
```

## *Execute\_Statement*

```
procedure_body Execute_Statement (  
    preserves Statement& s,  
    preserves Program& p  
)  
{  
  
  
  
  
  
  
  
  
  
}
```

## *BL Program Execution*

- Execution of a BL program as Program object seems simple when done as one or more recursive operations cooperating to execute the whole program
- What about executing the program one "step" at a time?

## *Program Execution Continued...*

- The problem seems to be related to the hierarchical, nested structure of Program objects
- It is difficult to stop and resume execution, and to keep track of a specific "position" in the program
- How can we execute a BL program in steps?

## *Code Generation*

- Translate program to "linear" (i.e., non-nested) structure, e.g., string of low-level (assembly-like) instructions
  - Primitive instructions are translated directly
  - Control structures are replaced with jump instructions
  - User-defined instructions are inlined

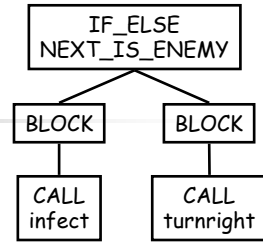
## An Example

```

IF next-is-enemy THEN
  infect
ELSE
  turnright
END IF
...

```

0	JUMP_IF_NOT_NEXT_IS_ENEMY
1	5
2	INFECT
3	JUMP
4	6
5	TURNRIGHT
6	...



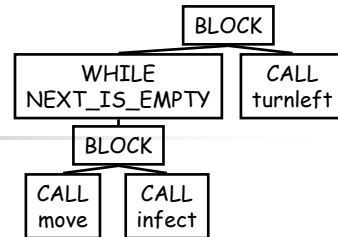
## Another Example

```

WHILE next-is-empty DO
  move
  infect
END WHILE
turnleft
...

```

0	JUMP_IF_NOT_NEXT_IS_EMPTY
1	6
2	MOVE
3	INFECT
4	JUMP
5	0
6	TURNLEFT
7	...





## *BugsWorld Virtual Machine*

---

- Memory
- Instruction set (target language)
- Program counter



## *Target Language*

---

- Primitive instructions
- Unconditional jump
- Conditional jumps



## *Primitive Instructions*

---

- MOVE
- TURNLEFT
- TURNRIGHT
- INFECT
- SKIP
- HALT

## *Unconditional Jump*

- JUMP addr



## *Conditional Jumps*

---

- JUMP\_IF\_NOT\_NEXT\_IS\_EMPTY addr
- JUMP\_IF\_NOT\_NEXT\_IS\_NOT\_EMPTY addr
- JUMP\_IF\_NOT\_NEXT\_IS\_ENEMY addr
- JUMP\_IF\_NOT\_NEXT\_IS\_NOT\_ENEMY addr
- JUMP\_IF\_NOT\_NEXT\_IS\_FRIEND addr
- JUMP\_IF\_NOT\_NEXT\_IS\_NOT\_FRIEND addr
- JUMP\_IF\_NOT\_NEXT\_IS\_WALL addr
- JUMP\_IF\_NOT\_NEXT\_IS\_NOT\_WALL addr
- JUMP\_IF\_NOT\_RANDOM addr
- JUMP\_IF\_NOT\_TRUE addr

## Handling IF Statements

...  
 IF condition THEN  
   block  
 END IF  
 ...

k-1	...
k	JUMP_IF_NOT_condition
k+1	k+n+2
k+2	block ↓
k+3	
...	
k+n+1	
k+n+2	...

## Handling IF\_ELSE Statements

...  
 IF condition THEN  
   if\_block  
 ELSE  
   else\_block  
 END IF  
 ...

k-1	...
k	JUMP_IF_NOT_condition
k+1	k+n1+4
k+2	if_block ↓
...	
k+n1+2	
k+n1+3	JUMP
k+n1+4	k+n1+n2+4 else_block ↓
...	
k+n1+n2+4	

## Handling WHILE Statements

...  
 WHILE condition DO  
   block  
 END WHILE  
 ...

k-1	...
k	JUMP_IF_NOT_condition
k+1	k+n+4
k+2	block ↓
k+3	
...	
k+n+1	
k+n+2	JUMP
k+n+3	k
k+n+4	...

## Handling CALL Statements

INSTRUCTION my-instruction IS  
   block  
 END my-instruction

...  
 my-instruction  
 ...

k-1	...
k	block ↓
k+1	
...	
k+n-1	
k+n	...