Compilation

Making code look worse to humans but better to processors

The goal of a compiler

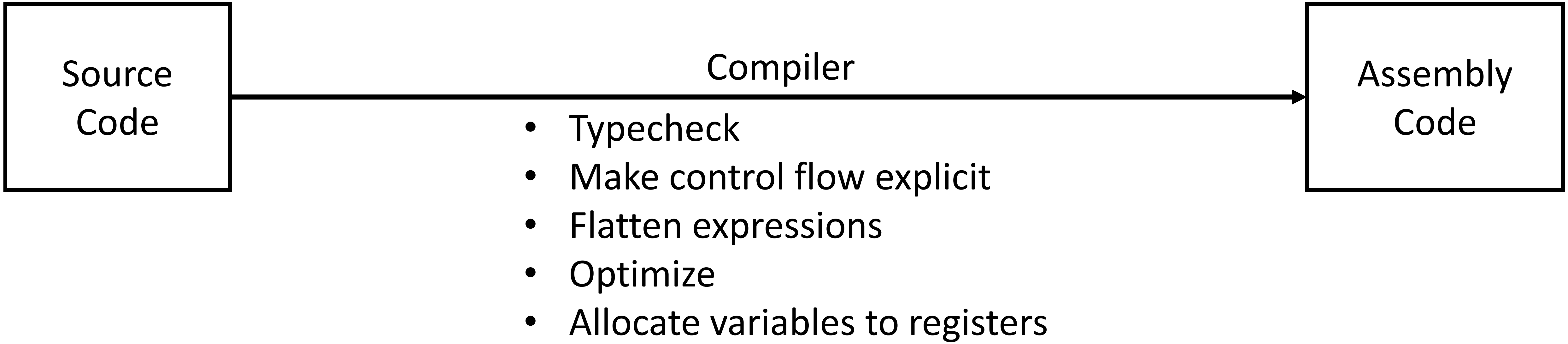
Translate from a **source language** to a **target language**

Usually a high-level language

- C
- SML
- Java
- Go
- Rust

Usually a low-level language

- X86 Assembly
- Java Bytecode
- C



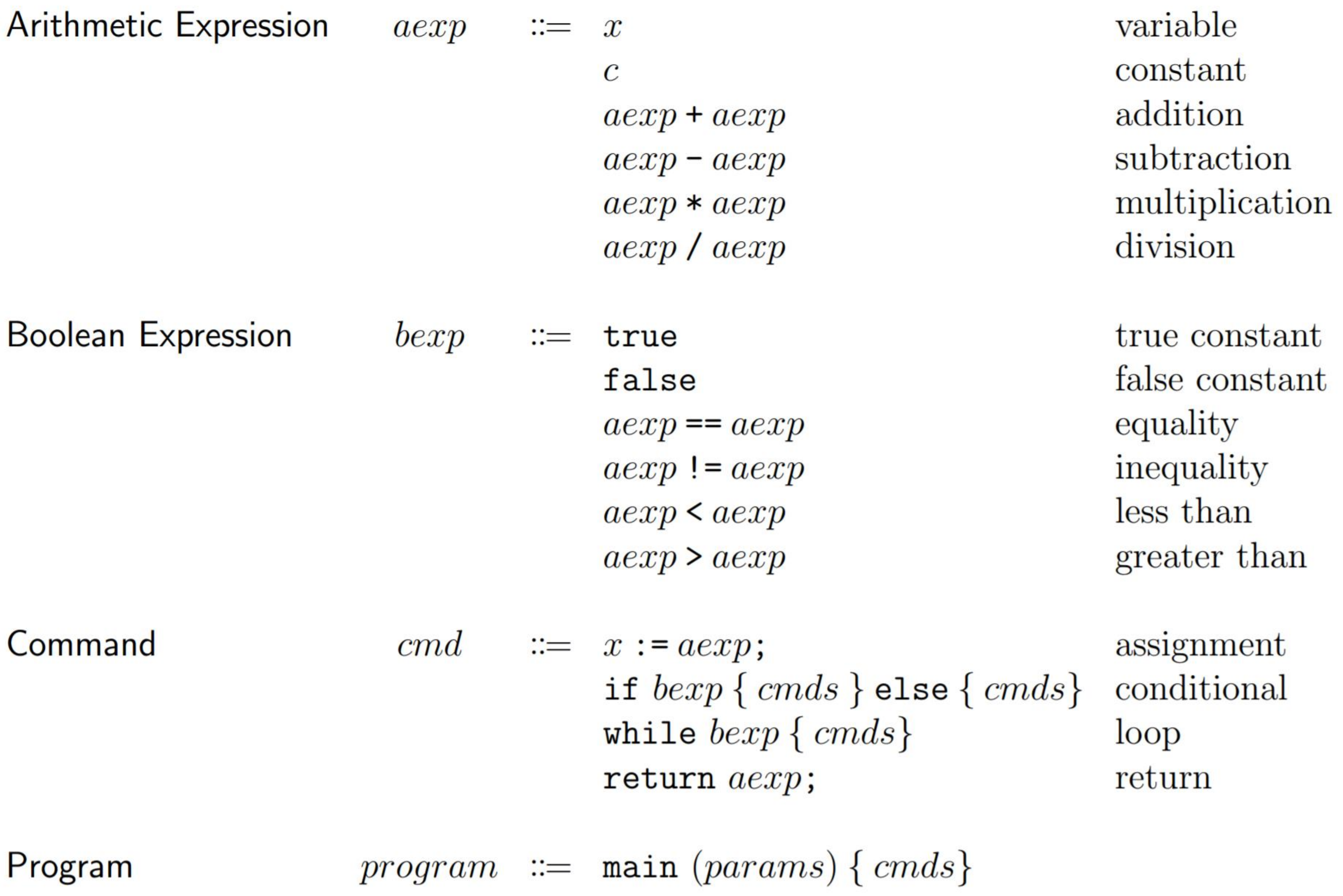
Source Code

Compiler

Assembly Code

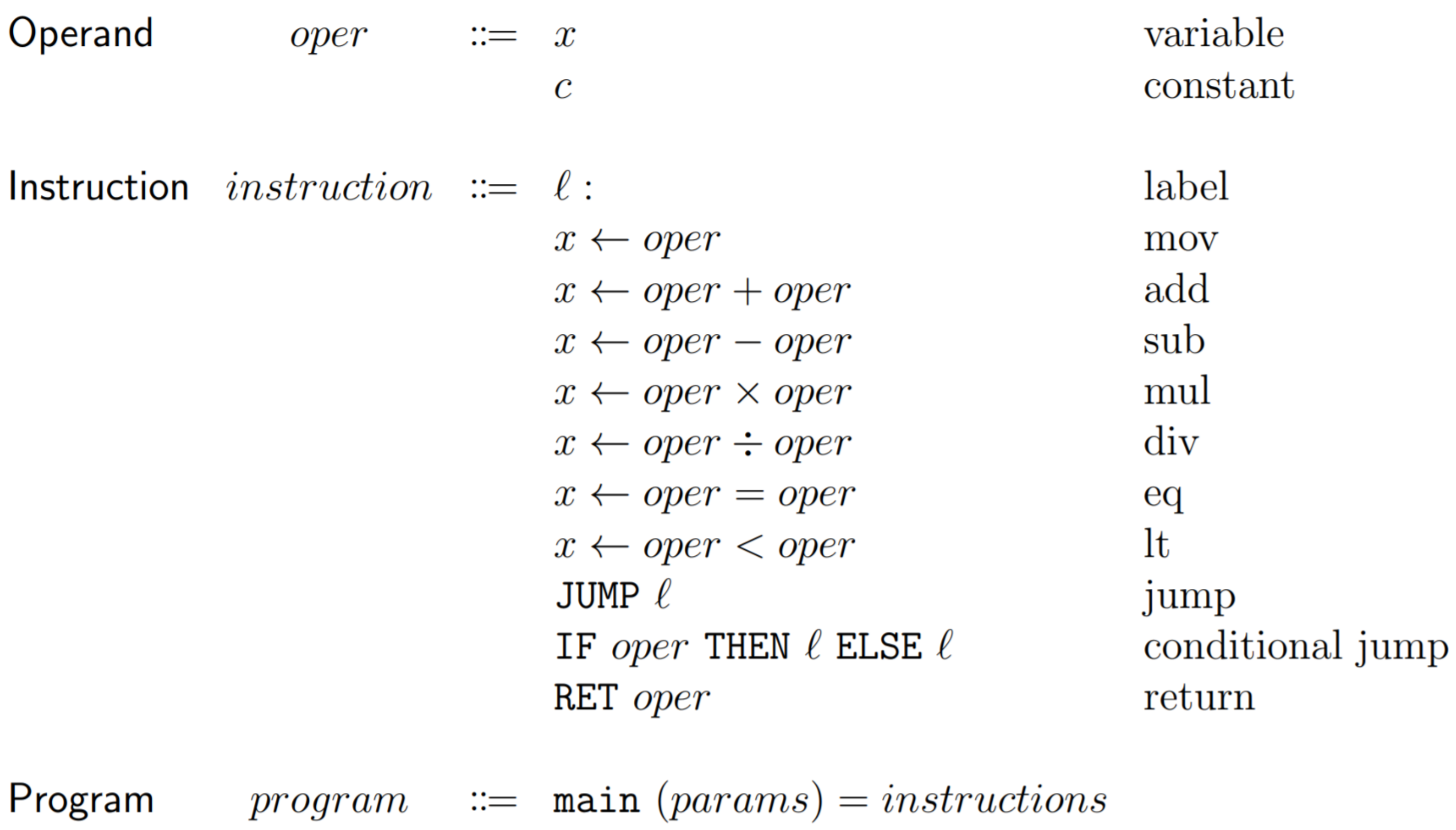
- Typecheck
- Make control flow explicit
- Flatten expressions
- Optimize
- Allocate variables to registers

Example Source Language: C--

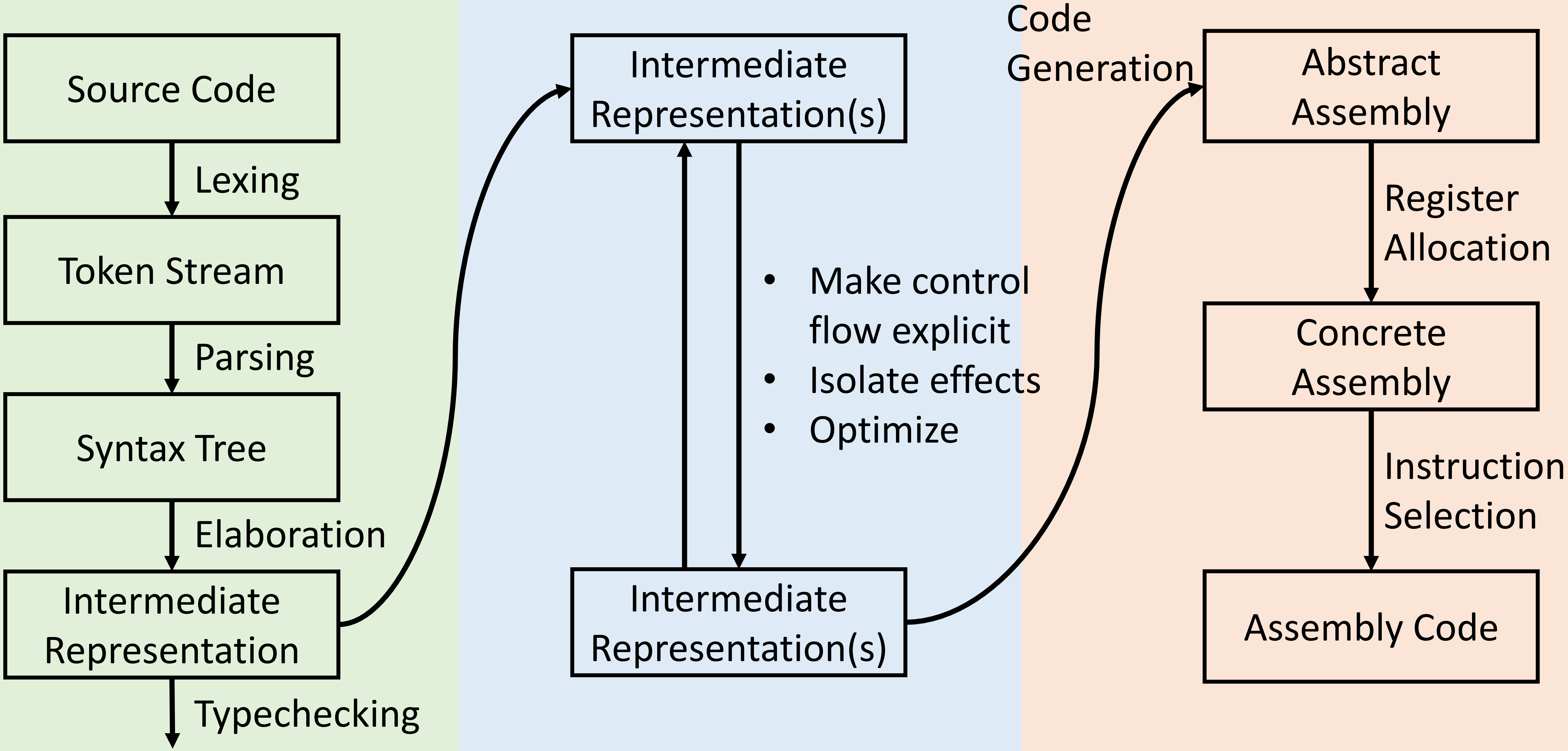


Arithmetic Expression	*aexp*	::=	*x*	variable
			c	constant
			aexp + *aexp*	addition
			aexp - *aexp*	subtraction
			aexp * *aexp*	multiplication
			aexp / *aexp*	division
Boolean Expression	*bexp*	::=	`true`	true constant
			`false`	false constant
			aexp == *aexp*	equality
			aexp != *aexp*	inequality
			aexp < *aexp*	less than
			aexp > *aexp*	greater than
Command	*cmd*	::=	*x* := *aexp*;	assignment
			`if` *bexp* { *cmds* } `else` { *cmds*}	conditional
			`while` *bexp* { *cmds*}	loop
			`return` *aexp*;	return
Program	*program*	::=	`main` (*params*) { *cmds*}	

Example (Abstract) Assembly Language



Operand	*oper*	::=	x	variable
			c	constant
Instruction	*instruction*	::=	$\ell :$	label
			$x \leftarrow oper$	mov
			$x \leftarrow oper + oper$	add
			$x \leftarrow oper - oper$	sub
			$x \leftarrow oper \times oper$	mul
			$x \leftarrow oper \div oper$	div
			$x \leftarrow oper = oper$	eq
			$x \leftarrow oper < oper$	lt
			JUMP ℓ	jump
			IF *oper* THEN ℓ ELSE ℓ	conditional jump
			RET *oper*	return
Program	*program*	::=	main (*params*) = *instructions*	



Source Code

Lexing

Token Stream

Parsing

Syntax Tree

Elaboration

Intermediate Representation

Typechecking

Intermediate Representation(s)

- Make control flow explicit
- Isolate effects
- Optimize

Intermediate Representation(s)

Code Generation

Abstract Assembly

Register Allocation

Concrete Assembly

Instruction Selection

Assembly Code

FRONT

MIDDLE

BACK

Notes that may be useful for the homework

- Intermediate representations (especially pages 4-7) http://www.cs.cmu.edu/~janh/courses/411/17/lec/11-irtrees.pdf
- Code Generation (especially pages 1-6) http://www.cs.cmu.edu/~janh/courses/411/17/lec/02-instsel.pdf

You should do this homework if you're considering taking 15-411 (Compiler Design)

Take 15-411 (Compiler Design)

I’ll be a TA