Step5: Generating *Tiny* Assembly Code

- In the fifth step of this course project, you will implement a compiler pass that generates *Tiny*: a two-address assembly code
- We will use the string-matching instruction selection scheme studied in lecture 10: the compiler maps each instruction in the three-address code, in isolation, into an equivalent sequence of *tiny* instructions
  - Implementing peephole optimization is not required
- We will use a software simulator to execute and test the compiled *Tiny* programs
  - For simplicity, we will assume the simulator allows infinite number of physical registers

Instruction Selection Examples

Three-address code		*Tiny* code
READI A	→	sys readi A
STOREI $T1, A	→	move r1 , A
ADDI $T1, A, $T2	→	move r1 , r2 addi A , r2
GE $T1, $T2, L1	→	cmpi r1 , r2 jge L1

Translation Example 1

Micro Program	Three-Address IR code	Tiny Assembly Code
PROGRAM test BEGIN INT a; INT b; INT c; FUNCTION VOID main( ) BEGIN READ(a,b) ; c := 1 / a - b * 2 ; WRITE(c); END END	LABEL main READI a READI b DIVI 1 a $T1 MULTI b 2 $T2 SUBI $T1 $T2 $T3 STOREI $T3 c WRITEI c	var a var b var c label main sys readi a sys readi b move 1 r1 divi a r1 move b r2 muli 2 r2 move r1 r3 subi r2 r3 move r3 c sys writei c

Translation Example 2

Micro Program	Three-Address IR code	Tiny Assembly Code

PROGRAM test
BEGIN
INT a;
INT b;
 	
FUNCTION VOID main( )
BEGIN
READ(a) ;
IF (a > 0)
b := 1 ;
ELSE
b := 0 ;
ENDIF
WRITE(b);
END
END
	LABEL main
	READI a
	LE a 0 L1
	STOREI 1 b
	JUMP L2
	LABEL L1
	STOREI 0 b
	LABEL L2
	WRITEI b
		var a
		var b
		label main
		sys readi a
		move 0 r1
		cmpi a r1
		jle L1
		move 1 b
		jmp L2
		label L1
		move 0 b
		label L2
		sys writei b

How To Use *Tiny* Simulator?

- Find “Tiny Documentation” on the course webpage for information about:
  - Available *Tiny* instructions and their format
  - How to build the simulator
  - How to execute *Tiny* programs on the simulator

Step 5 Test Cases and Output

- Your compiler should now be complete: accepts input programs that are written in *MICRO* and produces a machine code program written in *Tiny*
- The course web page has a set of input *MICRO* code examples and the expected output *Tiny* code for each *MICRO* code example
- The simulator is also available, along with the excepted output by the simulator when executing the *Tiny* program of each *MICRO* code example
- Due: TBA