CS 410 | Compiler Construction | Phase 3: Code Generator

EXPECTATIONS

As a team, **review your minimal dialect** (as in Phase 1) and update it only when necessary. Then…

A **Create a code generator** for your minimal dialect. Your code generator should:

- Input a stream of atoms from the output of your parser (as in Phase 2) or from a user, file, or standard channel.
- Output a machine code program in binary to a user, file, or standard channel.
- Support the following architecture with 32 bit instructions in absolute addressing mode.
  - Optionally, support both absolute addressing mode and displacement addressing mode.

Mode	**Size**	**4 bits** (0 to 3)	**1 + 3 bits** (4 to 7)	**4 bits** (8 to 11)	**20 bits** (12 to 31)	
Absolute	32 bits	opcode (0 to 9)	0 + cmp (0 to 6)	r (0 to 15)	a (0 to 2^{20} – 1)	
Mode	**Size**	**4 bits**	**1 + 3 bits**	**4 bits**	**4 bits** (12 to 15)	**16 bits** (16 to 31)
Displacement	32 bits	opcode (0 to 9)	1 + cmp (8 to 14)	rx (0 to 15)	ry (0 to 15)	d (0 to 2^{16} – 1)

- Translate your intermediate code atoms to these machine code instructions:

Mnemonic	**Opcode**	**Semantics: Absolute Mode**				**Semantics: Displacement Mode**		
CLR	**0**	fpreg[r] = 0				fpreg[rx] = 0		
ADD	**1**	fpreg[r] = fpreg[r] + memory[a]				fpreg[rx] = fpreg[rx] + memory[reg[ry] + d]		
SUB	**2**	fpreg[r] = fpreg[r] – memory[a]				fpreg[rx] = fpreg[rx] – memory[reg[ry] + d]		
MUL	**3**	fpreg[r] = fpreg[r] × memory[a]				fpreg[rx] = fpreg[rx] × memory[reg[ry] + d]		
DIV	**4**	fpreg[r] = fpreg[r] ÷ memory[a]				fpreg[rx] = fpreg[rx] ÷ memory[reg[ry] + d]		
JMP	**5**	if flag is true, then pc = a, where pc is reg[1]				if flag is true, then pc = reg[ry] + d, where pc is reg[1]		
CMP	**6**	flag = fpreg[r] cmp memory[a], where the result is true when…				flag = fpreg[rx] cmp memory[reg[ry] + d], where the result is true when…		
		cmp = 0	cmp = 1	cmp = 2	cmp = 3	cmp = 4	cmp = 5	cmp = 6
		always true	equal	lesser	greater	lesser or equal	greater or equal	unequal
LOD	**7**	fpreg[r] = memory[a]				fpreg[rx] = memory[reg[ry] + d]		
STO	**8**	memory[a] = fpreg[r]				memory[reg[ry] + d] = fpreg[rx]		
HLT	**9**	halt processor (terminate execution)				halt processor (terminate execution)		

- Be hand-written in Java, not be made by a compiler generator (which is reserved for a future assignment).
- Observe that the code generator just outputs machine code, whereas a target computer executes machine code.
- Observe that steps A to C are deeply interconnected. They are separate responsibilities, not separate deliverables.

B **Design both a label table and a first pass** for your code generator. Your design should:

- Build a label table while your code generator translates label-based atoms to placeholder instructions.
- Follow either the single-pass strategy or the multi-pass strategy.
- Incorporate your design into the code generator of step A above.
- Observe that steps A to C are deeply connected. They are separate responsibilities, not separate deliverables.

C **Design either a fixup table or a second pass** for your code generator. You design should:

- Apply the label table to finalize your label-based placeholder instructions.
- Follow either the single-pass strategy with a fixup table or the multi-pass strategy without a fixup table.
- Incorporate your design into the label table of step B above.
- Observe that steps A to C are deeply connected. They are separate responsibilities, not separate deliverables.

ASSESSMENT

As a team, submit your compiler phase deliverables to the assignment submission page in eCampus.

- **Create and attach a credits table** based on the actual contributions within your team as a PDF file.
  - For each of the lettered steps, credit up to half of the team as authors and up to half of the team as reviewers.
  - For each team member, credit them as either author, reviewer, or leave them uncredited on each step.
  - In the case of inequitable contributions within your team, notify the instructor in writing before or after submission.
- **Create and attach a disclosure** of met and unmet expectations as a PDF file.
  - If you know some of the expectations are unmet, list them.
  - If you believe all or the rest of the expectations are met, state that.
- **Attach your code generator** as one or more source code files.
- **Verify your submission** with the instructor before the compiler phase is due.