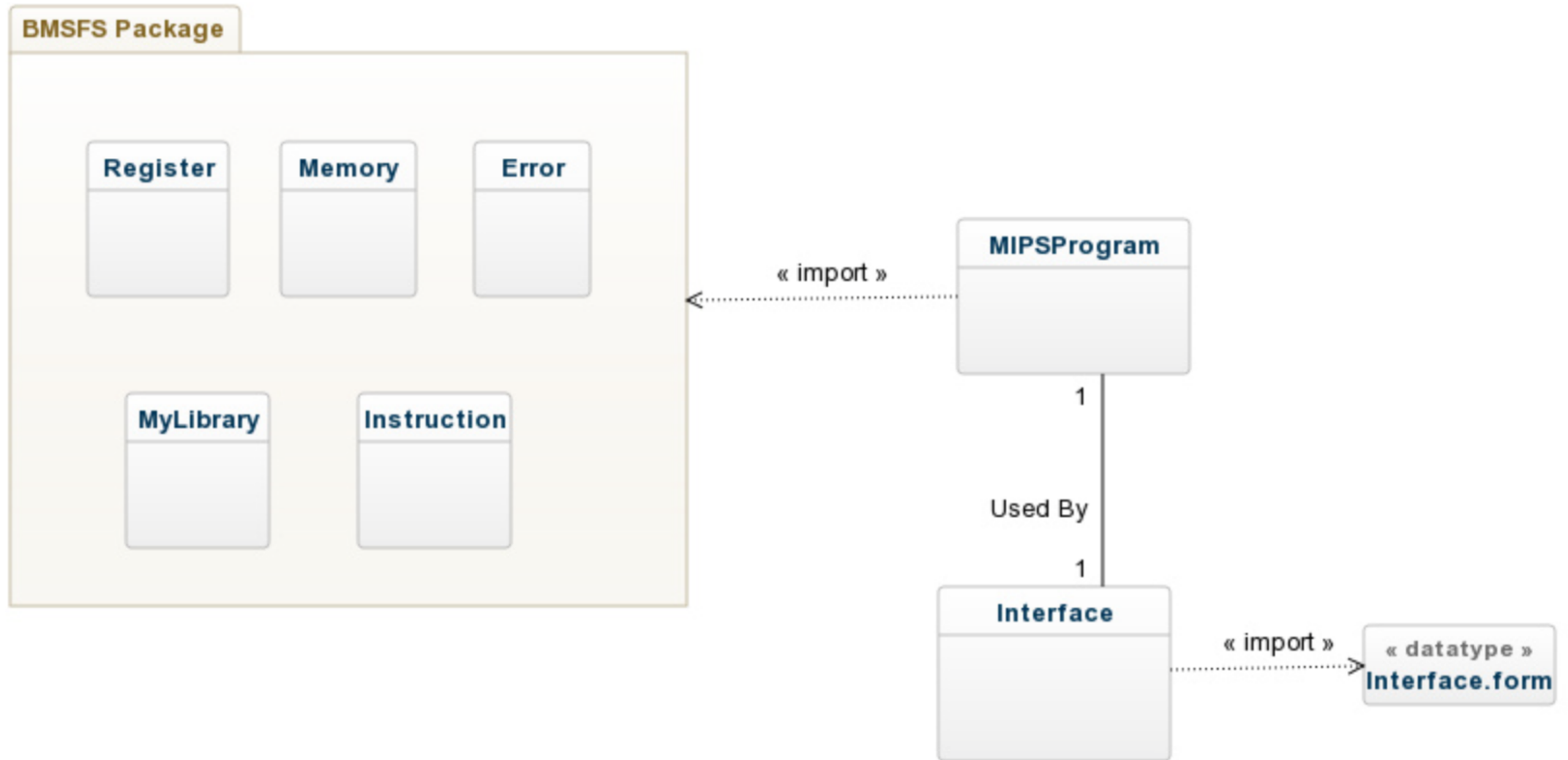


Code Report

Experiment - MIPS Assembly Language II
Team 9

Introduction

- MIPS II - A Virtual Lab experiment.
- Aim: Online/Browser-based experiment to aid learning
- Scale: 1 core module, 1 interface, 7 classes and over 2700 LOC
- Problems:
 - Java Plug-in required.
 - Use of legacy/deprecated functions
 - No documentation



Metrics (I)

Class/Metric	Functions/ class	Parameters/Fn	Statements/Fn	Cyclometric	LOC	LOC-Comment ratio
Memory.js	12	1	8	3	205	25.25
Error.js	7	1	10	1	50	25.0
MyLibrary.js	12	1	4	1	35	17.5
Register.js	14	1	2	1	156	32.0
MIPSProgram.js	26	0.5	8	3	432	25.82

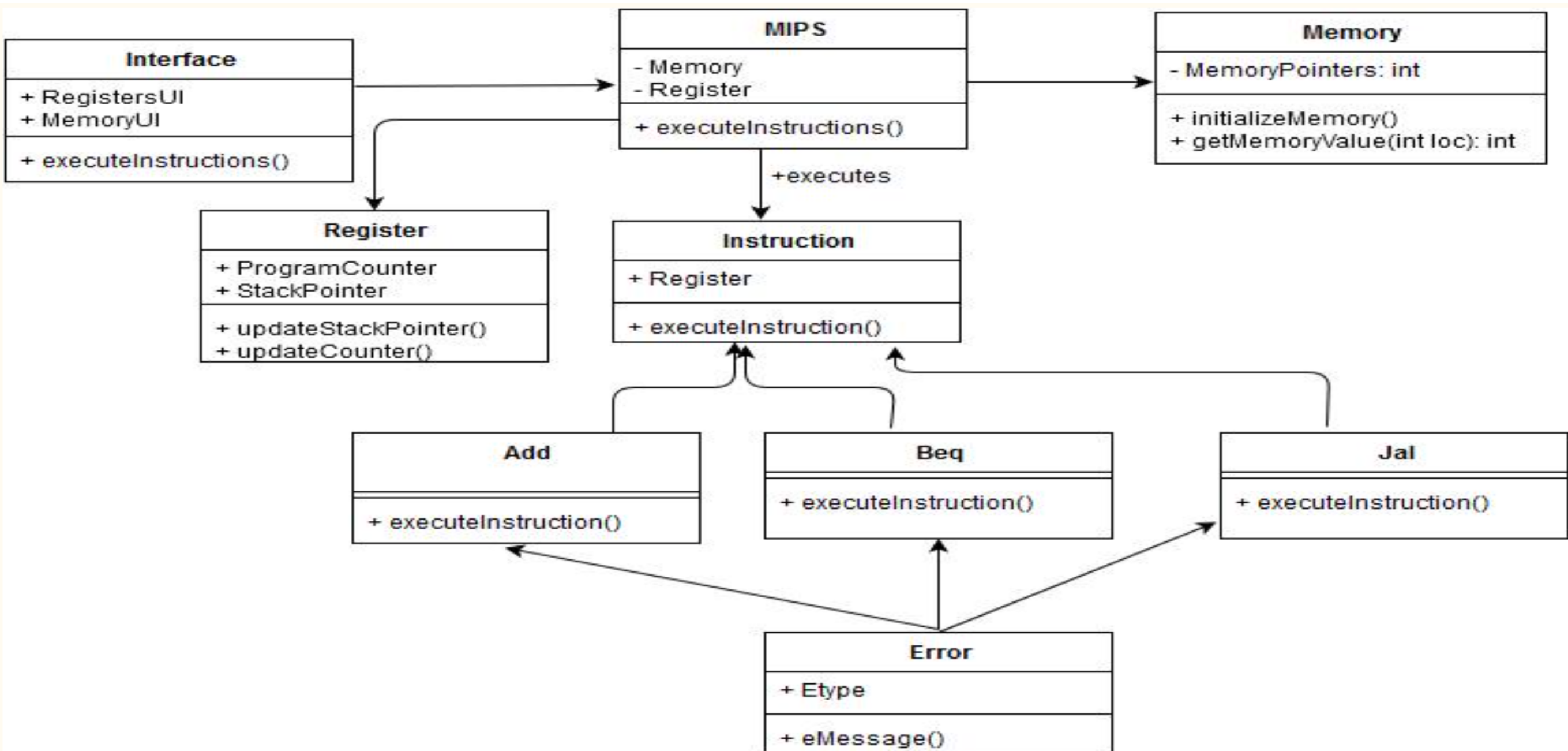
Code Smells

- Error - Divergent change, Switch statements, Lazy class, Shotgun surgery.
- Instruction - Conditional Complexity, Combinatorial Explosion,
- Memory - Dead code, Methods too big, Hardcode
- Register - Conditional Complexity, No comments.
- MyLibrary - Class/Variables names, Scope too wide, Inconsistent method/argument conventions, Incomplete Library Class.

Code Smells (II)

- Combinatorial Explosion
- Indecent Exposure:
- Message Chains
- Solution Sprawl

```
300     err = parseBasicInstrucction(str, lineNoInSource);
301 } else if (mnemonic.equalsIgnoreCase("addi")) {
302     err = parseBasicInstrucction(str, lineNoInSource);
303 } else if (mnemonic.equalsIgnoreCase("addiu")) {
304     err = parseBasicInstrucction(str, lineNoInSource);
305 } else if (mnemonic.equalsIgnoreCase("addu")) {
306     err = parseBasicInstrucction(str, lineNoInSource);
307 } else if (mnemonic.equalsIgnoreCase("and")) {
308     err = parseBasicInstrucction(str, lineNoInSource);
309 } else if (mnemonic.equalsIgnoreCase("andi")) {
310     err = parseBasicInstrucction(str, lineNoInSource);
311 } else if (mnemonic.equalsIgnoreCase("b")) {
312     err = parseBasicInstrucction(str, lineNoInSource);
313 } else if (mnemonic.equalsIgnoreCase("beq")) {
314     err = parseBasicInstrucction(str, lineNoInSource);
315 } else if (mnemonic.equalsIgnoreCase("beqz")) {
316     err = parseBasicInstrucction(str, lineNoInSource);
317 } else if (mnemonic.equalsIgnoreCase("bge")) {
318     err = parseBasicInstrucction(str, lineNoInSource);
319 } else if (mnemonic.equalsIgnoreCase("bgez")) {
320     err = parseBasicInstrucction(str, lineNoInSource);
321 } else if (mnemonic.equalsIgnoreCase("bgezal")) {
322     err = parseBasicInstrucction(str, lineNoInSource);
323 } else if (mnemonic.equalsIgnoreCase("bgt")) {
324     err = parseBasicInstrucction(str, lineNoInSource);
325 } else if (mnemonic.equalsIgnoreCase("bgtz")) {
326     err = parseBasicInstrucction(str, lineNoInSource);
327 } else if (mnemonic.equalsIgnoreCase("ble")) {
328     err = parseBasicInstrucction(str, lineNoInSource);
329 } else if (mnemonic.equalsIgnoreCase("blez")) {
330     err = parseBasicInstrucction(str, lineNoInSource);
331 } else if (mnemonic.equalsIgnoreCase("blt")) {
332     err = parseBasicInstrucction(str, lineNoInSource);
333 } else if (mnemonic.equalsIgnoreCase("bltz")) {
334     err = parseBasicInstrucction(str, lineNoInSource);
335 } else if (mnemonic.equalsIgnoreCase("bltzal")) {
336     err = parseBasicInstrucction(str, lineNoInSource);
337 } else if (mnemonic.equalsIgnoreCase("bne")) {
338     err = parseBasicInstrucction(str, lineNoInSource);
339 } else if (mnemonic.equalsIgnoreCase("bnez")) {
340     err = parseBasicInstrucction(str, lineNoInSource);
341 } else if (mnemonic.equalsIgnoreCase("clo")) {
342     err = parseBasicInstrucction(str, lineNoInSource);
343 } else if (mnemonic.equalsIgnoreCase("clz")) {
344     err = parseBasicInstrucction(str, lineNoInSource);
345 } else if (mnemonic.equalsIgnoreCase("div")) {
346     err = parseBasicInstrucction(str, lineNoInSource);
347 } else if (mnemonic.equalsIgnoreCase("divu")) {
348     err = parseBasicInstrucction(str, lineNoInSource);
349 } else if (mnemonic.equalsIgnoreCase("j")) {
350     err = parseBasicInstrucction(str, lineNoInSource);
351 } else if (mnemonic.equalsIgnoreCase("jal")) {
352     err = parseBasicInstrucction(str, lineNoInSource);
353 } else if (mnemonic.equalsIgnoreCase("jalr")) {
354     err = parseBasicInstrucction(str, lineNoInSource);
355 } else if (mnemonic.equalsIgnoreCase("jr")) {
356     err = parseBasicInstrucction(str, lineNoInSource);
357 } else if (mnemonic.equalsIgnoreCase("la")) { // CHECK
358     err = parseLoadStoreInstr(str, lineNoInSource);
359 } else if (mnemonic.equalsIgnoreCase("lb")) {
360     err = parseLoadStoreInstr(str, lineNoInSource);
361 } else if (mnemonic.equalsIgnoreCase("lh")) {
362     err = parseLoadStoreInstr(str, lineNoInSource);
363 } else if (mnemonic.equalsIgnoreCase("li")) {
364     err = parseBasicInstrucction(str, lineNoInSource);
365 } else if (mnemonic.equalsIgnoreCase("lw")) {
366     err = parseLoadStoreInstr(str, lineNoInSource);
367 } else if (mnemonic.equalsIgnoreCase("mfhi")) {
368     err = parseBasicInstrucction(str, lineNoInSource);
369 } else if (mnemonic.equalsIgnoreCase("mflo")) {
370     err = parseBasicInstrucction(str, lineNoInSource);
371 } else if (mnemonic.equalsIgnoreCase("mthi")) {
372     err = parseBasicInstrucction(str, lineNoInSource);
373 } else if (mnemonic.equalsIgnoreCase("mtlo")) {
374     err = parseBasicInstrucction(str, lineNoInSource);
375 } else if (mnemonic.equalsIgnoreCase("move")) {
376     err = parseBasicInstrucction(str, lineNoInSource);
377 } else if (mnemonic.equalsIgnoreCase("movn")) {
378     err = parseBasicInstrucction(str, lineNoInSource);
379 } else if (mnemonic.equalsIgnoreCase("movz")) {
380     err = parseBasicInstrucction(str, lineNoInSource);
381 } else if (mnemonic.equalsIgnoreCase("mul")) {
382     err = parseBasicInstrucction(str, lineNoInSource);
383 } else if (mnemonic.equalsIgnoreCase("mulo")) {
384     err = parseBasicInstrucction(str, lineNoInSource);
385 } else if (mnemonic.equalsIgnoreCase("mult")) {
386     err = parseBasicInstrucction(str, lineNoInSource);
387 } else if (mnemonic.equalsIgnoreCase("neg")) {
388     err = parseBasicInstrucction(str, lineNoInSource);
389 } else if (mnemonic.equalsIgnoreCase("nor")) {
390     err = parseBasicInstrucction(str, lineNoInSource);
391 } else if (mnemonic.equalsIgnoreCase("not")) {
392     err = parseBasicInstrucction(str, lineNoInSource);
393 } else if (mnemonic.equalsIgnoreCase("or")) {
394     err = parseBasicInstrucction(str, lineNoInSource);
395 } else if (mnemonic.equalsIgnoreCase("ori")) {
```



Proposed Structure

Design Patterns

- Builder pattern -

```
if (operands[numOperands - 1] == -1) { // invalid register  
    err = new Error("Syntax error @ line : " + lineNoInSource, Error.ASSEMBLE_TIME_ERROR);  
    return err;  
}
```

- Interpreter pattern - Instruction is used to interpret 'language' from GUI before processing.
- Observer pattern - GUI subscribes to changes in BFSFS objects.
- Chain of Responsibility - Instruction is used to comm. between GUI and core.
- Iterator - Implemented in Instruction module to execute one at a time.

Take-aways

- Time required !~ Lines of code
- 1) Think/Design 2) Code. Not the other way round.
- Need to have common coding standards when working in a team.
- No free lunch

“I'm not a great programmer; I'm just a good programmer with great habits.”

– Martin Fowler