Jacob Gidley

ID: 01350817

Assignment #1: Unit Tests

**Test #1**
Test to see if *add* works:

```
addimmediate r1 5 (91 05)
addimmediate r2 12 (92 0C)
add r1 r2 r3 (11 23)
interrupt 0 (8000)
```

Expected results:
R1 == 5
R2 == 12
R3 == 17

---

**Test #2**
Test to see if *subtract* works:

```
addimmediate r1 45 (91 2D)
addimmediate r2 9 (92 09)
subtract r1 r2 r3 (51 23)
interrupt 0 (8000)
```

Expected results:
R1 == 45
R2 == 9
R3 == 36

---

**Test #3**
Test to see if *multiply* works:

```
addimmediate r1 4 (91 04)
addimmediate r2 7 (92 07)
multiply r1 r2 r3 (41 23)
interrupt 0 (8000)
```

Expected results:
R1 == 4
R2 == 7
R3 == 28

---

**Test #4**
Test to see if *divide* works:

addimmediate r1 32 (91 20)
addimmediate r2 4 (92 04)
divide r1 r2 r3 (31 23)
interrupt 0 (8000)

Expected results:
R1 == 32
R2 == 4
R3 == 8

---

**Test #5**
Test to see if *and* works:

addimmediate r1 4 (91 04)
addimmediate r2 13 (92 0D)
and r1 r2 r3 (21 23)
interrupt 0 (8000)

Expected results:
R1 == 4
R2 == 13
R3 == 4

---

**Test #6**
Test to see if *or* works:

addimmediate r1 16 (91 10)
addimmediate r2 7 (92 07)
or r1 r2 r3 (61 23)
interrupt 0 (8000)

Expected results:
R1 == 16
R2 == 7
R3 == 23

---

**Test #7**
Test to see if *leftshift* works:

addimmediate r1 18 (91 12)
leftshift r1 4 (71 04)
interrupt 0 (8000)

**Expected result:**
R1 == 288

---

**Test #8**
Test to see if *rightshift* works:

addimmediate r1 56 (91 38)
rightshift r1 5 (71 25)
interrupt 0 (8000)

Expected result:
R1 == 1

---

**Test #9**
Test to see if *halt* works

addimmediate r1 17 (91 11)
addimmediate r2 3 (92 03)
add r1 r2 r2 (11 22)
interrupt 0 (8000)
halt (0000)
subtract r1 r2 r2 (51 22) *; The program should never get to this point because of the halt*
interrupt 0 (8000)

Expected results:
R1 == 17
R2 == 20

---

**Test #10**

Test to see if *branchifequal* works when two register values are equal:

addimmediate r1 3 (91 03)
addimmediate r2 3 (92 03)
branchifequal r1 r2 10 (A1 20 00 0A)
subtract r1 r2 r3 (51 23) *; This branch should not be taken*
interrupt 0 (8000)
halt (0000)
add r1 r2 r3 (11 23) *; This branch should be taken if r1 == r2*
interrupt 0 (8000)

Expected results:
R1 == 3
R2 == 3
R3 == 6

---

**Test #11**

Test to see if *branchifequal* works when the two register values are NOT equal:

addimmediate r1 15 (91 0F)
addimmediate r2 7 (92 07)
branchifequal r1 r2 10 (A1 20 00 0A)
subtract r1 r2 r3 (51 23) *; This branch should be taken*
interrupt 0 (8000)
halt (0000)
add r1 r2 r3 (11 23) *; This branch should not be taken*
interrupt 0 (8000)

Expected results:
R1 == 15
R2 == 7
R3 == 8

---

**Test #12**

Test to see if *branchifless* works when first register is less than the second:

addimmediate r1 6 (91 06)
addimmediate r2 8 (92 08)
branchifless r1 r2 8 (B1 20 00 08)
interrupt 0 (8000) *; This branch should not be taken*
halt (0000)
multiply r1 r2 r3 (41 23) *; This branch should be taken*
interrupt 0 (8000)

Expected results:
R1 == 6
R2 == 8
R3 == 48

---

**Test #13**
Test to see if *branchifless* works when first register is greater than the second:

addimmediate r1 8 (91 08)
addimmediate r2 6 (92 06)
branchifless r1 r2 4 (B1 20 00 08)
add r1 r2 r3 (11 23) *; This branch should be taken*
interrupt 0 (8000)
halt (0000)
multiply r1 r2 r3 (41 23) *; This branch should not be taken*
interrupt 0 (8000)

Expected results:
R1 == 8
R2 == 6
R3 == 14

**Test #14**
Test to see if *jump* works

addimmediate r1 8 (91 08)
addimmediate r2 5 (92 05)
jump 12 (C0 00 00 0C)
subtract r1 r2 r3
multiply r1 r2 r2
add r1 r2 r3 *; The jump should go to this instruction*
interrupt 0 (8000)

Expected results:
R1 == 8
R2 == 5
R3 == 13

---

**Test #15**

Test program that creates a loop that simulates multiplication using addition and a negative branch loop. This program will perform what 3 * 4 would do.

addimmediate r1 3 (91 03)
addimmediate r2 5 (92 05)
add r1 r3 r3 (11 33)   *; Add 3 into r3*
addimmediate r4 1 (94 01)   *; Add to counter*
branchifless r4 r2 -4 (B4 2F FF FC)
*; Go back to add instruction until r4 < r2 (4 loops)*
interrupt 0 (8000)

Expected results:
R1 == 3
R2 == 4
R3 == 12
R4 == 5

---

**Test #16**

Test to see if *load* and store works

addimmediate r1 5 (91 05)
addimmediate r2 25 (92 19)
store r1 r2 0 (F1 20)
load r3 r2 0 (E1 30)
interrupt 0 (8000)

Expected results:
R1 == 5
R2 == 25
R3 == 5

---

**Test #17**

Test a program that simulates division using subtraction and a jump loop:

addimmediate r1 27 (91 1B)
addimmediate r2 8 (92 08)
addimmediate r3 0 (93 00)   *; r3 is the quotient*
branchifless r1 r2 12 (B1 20 00 0C)   ; *Branch to end if r2 > r1*
subtract r1 r2 r1 (51 21)
addimmediate r3 1 (93 01)
jump 6 (C0 00 00 06) *; Jump up if r1 > 0*
interrupt 0 (8000)

Expected results: R1 == 3, R2 == 8, R3 == 3

---

**Test #18**
Test to see if *leftshift* and *rightshift* work correctly on the same number:

```
addimmediate r1 3 (91 03)
leftshift r1 4 (71 00 00 04)     ; r1 == 48
interrupt 0 (8000)
rightshift r1 2 (71 10 00 02)   ; r1 == 12
interrupt 0 (8000)
```

Expected results:
R1 == 48
R1 == 12

---

**Test #19**
Test a NOP equivalent in SIA:

```
addimmediate r1 1 (91 01)
addimmediate r1 0 (91 00)      ; NOP
interrupt 0 (8000)
```

Expected Results:
R1 == 1

---

**Test #20**
Test program for a load immediate command in SIA:

```
addimmediate r1 1 (91 01)      ; Set r1 with an initial value
sub r1 r1 r1      (51 11)
addimmediate r1 32 (91 20)
interrupt 0 (8000)
```

Expected Results:
R1 == 32

---

**Test #21**
Test program for a clear instruction:

addimmediate r5 40 (95 28)
interrupt 0 (8000)
sub r5 r5 r5 *; Clear value from r5*
interrupt 0 (8000)

Expected Results:
R5 == 40
R5 == 0

---

**Test #22**
Test program for a branch if greater than or equal instruction:

addimmediate r1 10 (91 0A)
addimmediate r2 08 (92 08)
addimmediate r1 -1 (91 FF) *; Decrement r1 by 1*
branchifless r2 r1 -2 (B2 1F FF FE)
branchifequal r1 r2 -6 (A1 2F FF FA)
interrupt 0 (8000)

Expected Results:
R1 == 7
R2 == 8

---

**Test #23**
Test program for a branch if less than or equal instruction:

addimmediate r1 4 (91 04)
addimmediate r2 6 (92 06)
addimmediate r1 1 (91 01) *; Increment r1 by 1*
branchifless r1 r2 -2 (B1 2F FF FE)
branchifequal r1 r2 -6 (A1 2F FF FA)
interrupt 0 (8000)

Expected Results:
R1 == 7
R2 == 6

---

**Test #24**
Test program for a branch if not equal instruction (1st register is less than the 2nd):

addimmediate r1 2 (91 02)
addimmediate r2 6 (92 06)
addimmediate r1 1 (91 01)     *; both branches should loop back to this instruction*
addimmediate r2 -1 (92 FF)
branchifless r1 r2 -4 (B1 2F FF FC)
branchifless r2 r1 -8 (B2 1F FF F8)
interrupt 0 (8000)

Expected Results:
R1 == 4
R2 == 4

---

**Test #25**
Test program for a branch if not equal instruction (1st register is greater than the 2nd):

addimmediate r1 8 (91 08)
addimmediate r2 4 (92 04)
addimmediate r1 -1 (91 FF)    *; both branches should loop back to this instruction*
addimmediate r2 1 (92 01)
branchifless r1 r2 -4 (B1 2F FF FC)
branchifless r2 r1 -8 (B2 1F FF F8)
interrupt 0 (8000)

Expected Results:
R1 == 6
R2 == 6

---

**Test #26**
Test what happens when dividing by zero:

addimmediate r1 2 (91 02)
divide r1 r0 r2 (31 02)
interrupt 0 (8000)

Expected Results:
- Program crashes and error will be handled by thrown exception in JVM

---

**<u>Test #27</u>**
Test what happens when jumping past the end of a program:

```
addimmediate r1 1 (91 01)
addimmediate r2 4 (92 04)
jump 14 (C0 00 00 0E)
add r1 r2 r3 (11 23)
interrupt 0 (8000)
```

Expected Results:
- Program halts

---

**<u>Test #28</u>**
Test program to see if iterateover works by creating a linked list and summing up all the values in it. The starting address of the list will be 96.

```
addimmediate r1 96 (91 60)   ; 1st value in linked list
addimmediate r2 6 (92 06)    ; value of 6
store r1 r2 0    (F1 20)     ; store in r1 what's in r2 (6)
sub r1 r1 r1     (51 11)     ; clear r1
sub r2 r2 r2     (52 22)     ; clear r2

addimmediate r1 100 (91 64) ; next node in list
addimmediate r2 60 (92 3C)
store r1 r2 0 (F1 20)
sub r1 r1 r1 (51 11)
sub r2 r2 r2 (52 22)

addimmediate r1 104 (91 68) ; 2nd value in linked list
addimmediate r2 7 (92 07)    ; value of 7
store r1 r2 0 (F1 20)
sub r1 r1 r1 (51 11)
sub r2 r2 r2 (52 22)

addimmediate r1 108 (91 6C) ; next node in list (null)
addimmediate r2 0 (92 00)
store r1 r2 0 (F1 20)
sub r1 r1 r1 (51 11)
sub r2 r2 r2 (52 22)

addimmediate r1 96 (91 60)   ; Set r1 to start address of linked list
load r2 r1 0 (E2 10)     ; The iterateover instruction should jump back to this line
add r2 r3 r3 (12 33)     ; Add to the sum of the list
iterateover r1 4 4 (D1 04 00 04)
interrupt 0 (8000)
```

Expected Results:
R1 == 96
R2 == 7 *; Last value put in r2*
R3 == 13 *; The sum of the two nodes in the list*

---

**Test 29:**
Test to see what happens when memory is printed:

addimmediate r1 4 (91 04)
addimmediate r2 20 (92 14)
store r1 r2 0 (F1 20)
load r3 r2 0 (E3 20)
interrupt 1 (8001) *; Print all memory*

Expected Results:
- All memory is printed. I don't know what the values of the first 19 addresses would be.

| Memory address | Value |
|---|---|
| … | … |
| … | … |
| 20 - | 4 |

---

**Test 30:**
Test to see if what happens when no instructions are given:

Expected Results:
- No input should cause the program to end normally.

---

**Test 31:**
Test to see if subtracting a negative number from a positive number yields an addition result:

addimmediate r1 4 (91 04)
addimmediate r2 -9 (92 F7)
subtract r1 r2 r3 (51 23)
interrupt 0 (8000)

Expected Results:
R1 == 4
R2 == -9
R3 == 13

---

**<u>Test 32:</u>**
Test to see if adding a negative number to a positive number yields a subtraction result:

addimmediate r1 7 (91 07)
addimmediate r2 -12 (92 F4)
add r1 r2 r3 (11 23)
interrupt 0 (8000)

Expected Results:
R1 == 7
R2 == -12
R3 == -5

---

**<u>Test 33:</u>**
Test to see if an upward jump works with a branch statement and an infinite loop doesn't occur as a result:

addimmediate r1 10 (91 0A)
addimmediate r2 -3 (92 FD)
multiply r1 r2 r3 (41 23) *; The jump should loop here*
add r3 r4 r4 (13 44)
addimmediate r2 1 (92 01)
branchifequal r2 r5 10 (A2 50 00 0A) *; Branch over the jump when r2 == 0*
jump 4 (C0 00 00 04)
interrupt 0 (8000)

Expected Results:
R1 == 10
R2 == 0
R3 == -10
R4 == -60
R5 == 0

---

**<u>Test 34:</u>**
Test to see if multiplying by zero will result in zero:

addimmediate r1 6 (91 06)
addimmediate r2 3 (92 03) *; Assign this register a value. Should == 0 at end of program*
multiply r1 r0 r2 (41 02) *; Multiply r1 by r3, value of r2 should now == 0*
interrupt 0 (8000)

Expected Results:
R0 == 0
R1 == 6, R2 == 0

**<u>Test 35:</u>**
Test to see if adding by zero will result in an unchanged result:

```
addimmediate r1 28 (91 1C)
addimmediate r2 6 (92 06)      ; Assign this register a value. Should == 28 at end of program
add r1 r0 r2 (11 02)     ; Add 0 to register 1, value of r2 should be == 28
interrupt 0 (8000)
```

Expected Results:
R0 == 0
R1 == 28
R2 == 28

---

**<u>Test 36:</u>**
Test to see if subtracting by zero will result in an unchanged value:

```
addimmediate r1 16 (91 10)
addimmediate r2 9 (92 09)      ; Assign this register a value. Should == 16 at end of program
subtract r1 r0 r2 (51 02)      ; Subtract 0 from register 1, value of r2 should be == 16
interrupt 0 (8000)
```

Expected Results:
R0 == 0
R1 == 16
R2 == 16

---

**<u>Test 37:</u>**
Test to see if multiplying by a negative number will result in a negative value:

```
addimmediate r1 15 (91 0F)
addimmediate r2 -3 (92 FD)
addimmediate r3 1 (93 01)      ; This value should be negative @ end of program
multiply r1 r2 r3 (41 23)      ; Multiply r1 by r2
interrupt 0 (8000)
```

Expected Results:
R1 == 15
R2 == -3
R3 == -45

---

**Test 38:**

Test to see if dividing by a negative number will result in a negative value:

```
addimmediate r1 21 (91 15)
addimmediate r2 -7 (92 F9)
addimmediate r3 1 (93 01)      ; This value should be negative @ end of program
divide r1 r2 r3 (31 23)        ; Divide r1 by r2
interrupt 0 (8000)
```

Expected Results:
R1 == 21
R2 == -7
R3 == -3

---

**Test 39:**

Test to see if multiplying by a large number will yield correct results:

```
addimmediate r1 127 (91 7F)
addimmediate r2 1 (92 01)
addimmediate r3 2 (93 02)
addimmediate r4 3 (94 03)
multiply r1 r1 r2 (41 12)      ; Multiply r1 by r1 (127 * 127)
multiply r1 r2 r3 (41 23)      ; 127 * 16,129
multiply r2 r3 r4 (42 34)      ; 16,129 * 2,048,383
interrupt 0 (8000)
```

Expected Results:
R1 == 127
R2 == 16,129
R3 == 2,048,383
R4 == A value that will result in an overflow and will be handled by an exception in the JVM.

---

**Test 40:**

Test to see if dividing a small number by a large number will yield a result of zero:

```
addimmediate r1 4 (91 04)
addimmediate r2 16 (92 10)
addimmediate r3 2 (93 02)      ; This value should be 0 @ end of program
divide r1 r2 r3 (31 23)        ; Divide r1 by r2
```

Expected Results:
R1 == 4
R2 == 16
R3 == 0