CSE 140 - Computer Archictecture
Project #1

MIPS Instruction Interpretation in *C*

Christopher DeSoto & Aleksandr Brodskiy

March 9, 2018

The primary testing strategy for the implementation of a *C* instruction set interpreter for the MIPS assembly language was to develop test cases, six in total, essentially contingent on the format of the provided TEST(1−6).S files located in the TEST_CASES directory. These test cases were set−up as follows:
Test Case 1:

```
# Short test case for your project.
#
# Note that this is by no means a comprehensive test!
#

# Accessing invalid non-word aligned memory address
# resulting in memory exception print statement and
# termination.

                .text
                addiu   $a0,$0,3
                addiu   $a1,$0,2
                lui     $t0,64
                addiu   $t0,$t0,8192
                sw      $a1 0($t0)      # Accessing valid memory address
                lw      $a2 0($t0)
                addiu   $a1,$a1,2
                addiu   $t0,$t0,-1
                sw      $a1 0($t0)
```

Test Case 2:

```
# Short test case for your project.
#
# Note that this is by no means a comprehensive test!
#

# Accessing invalid out-of-range memory
# address resulting in memory exception
# print statement and termination.

                .text
                addiu   $a0,$0,3
                addiu   $a1,$0,2
                lui     $t0,64
                addiu   $t0,$t0,8192
                sw      $a1 0($t0)      # Accessing valid memory address
                lw      $a2 0($t0)
                addiu   $a1,$a1,2
                addiu   $t0,$t0,8192
                sw      $a1 0($t0)
```

Test Case 3:

```
# Short test case for your project.
#
# Note that this is by no means a comprehensive test!
#

# Accessing invalid out-of-range memory
# address resulting in memory exception
# print statement and termination.

                .text
                addiu   $a0,$0,3
                addiu   $a1,$0,2
                lui     $t0,64
                addiu   $t0,$t0,8192
                sw      $a1 0($t0)      # Accessing valid memory address
                lw      $a2 0($t0)
                addiu   $a1,$a1,2
                addiu   $t0,$t0,-8192
                sw      $a1 0($t0)
```

Test Case 4:

```
# Short test case for your project.
#
# Note that this is by no means a comprehensive test!
#

# Accessing valid memory address then
# runs an unsupported instruction and
# terminates

                .text
                addiu   $a0,$0,3
                addiu   $a1,$0,2
                lui     $t0,64
                addiu   $t0,$t0,8192
                sw      $a1 0($t0)
                lw      $a2 0($t0)
                addiu   $a1,$a1,2
                addi    $t0,$t0,-8192   # Unsupported instruction. Exit(0)
                sw      $a1 0($t0)
```

Test Case 5:

```
# Short test case for your project.
#
# Note that this is by no means a comprehensive test!
#

# Sums digits from 100 -> 0 and
# tests the lw & sw instructions
# at the first available memory
# address.

                .text
_start:
                addiu   $t0,$0,100
                addiu   $t1,$0,0
                addiu   $t2,$0,0
loop:           addu    $t1,$t1,$t0
                addiu   $t0,$t0,-1
                bne     $t0,$0,loop
                lui     $t0,0x0040
                addiu   $t0,$t0,0x1000
                sw      $t1,0($t0)
                lw      $t2,0($t0)
                sll     $t2,$t2,2=
```

Test Case 6:

```
# Short test case for your project.
#
# Note that this is by no means a comprehensive test!
#

# If $zero can be written to, code
# results in an infinite loop.

                .text
_start:
                addiu   $8, $0, 0
test:   addiu   $0, $0, 10
                bne     $0, $8 test
```

In this manner it is observable that the test case utilize a variety of procedural programming techniques, primarily loop iteration, jumps in memory referencing, and routine calls. As such the output for the cases are as follows:
Output 1:

```
Executing instruction at 00400000: 24040003
addiu   $4, $0, 3
New pc = 00400004
Updated r04 to 00000003
No memory location was updated.
Executing instruction at 00400004: 24050002
addiu   $5, $0, 2
New pc = 00400008
Updated r05 to 00000002
No memory location was updated.
Executing instruction at 00400008: 3c080040
lui     $8, 0x40
New pc = 0040000c
Updated r08 to 00400000
No memory location was updated.
Executing instruction at 0040000c: 25082000
addiu   $8, $8, 8192
New pc = 00400010
Updated r08 to 00402000
No memory location was updated.
Executing instruction at 00400010: ad050000
sw      $5, 0($8)
New pc = 00400014
No register was updated.
Updated memory at address 00402000 to 00000002
Executing instruction at 00400014: 8d060000
lw      $6, 0($8)
New pc = 00400018
Updated r06 to 00000002
No memory location was updated.
Executing instruction at 00400018: 24a50002
addiu   $5, $5, 2
New pc = 0040001c
Updated r05 to 00000004
No memory location was updated.
Executing instruction at 0040001c: 2508ffff
addiu   $8, $8, -1
New pc = 00400020
Updated r08 to 00401fff
No memory location was updated.
Executing instruction at 00400020: ad050000
sw      $5, 0($8)
Memory Access Exception at 0x00400020: address 0x00401fff
```

Output 2:

```
Executing instruction at 00400000: 24040003
addiu   $4, $0, 3
New pc = 00400004
Updated r04 to 00000003
No memory location was updated.
Executing instruction at 00400004: 24050002
addiu   $5, $0, 2
New pc = 00400008
Updated r05 to 00000002
No memory location was updated.
Executing instruction at 00400008: 3c080040
lui     $8, 0x40
New pc = 0040000c
Updated r08 to 00400000
No memory location was updated.
Executing instruction at 0040000c: 25082000
addiu   $8, $8, 8192
New pc = 00400010
Updated r08 to 00402000
No memory location was updated.
Executing instruction at 00400010: ad050000
sw      $5, 0($8)
New pc = 00400014
No register was updated.
Updated memory at address 00402000 to 00000002
Executing instruction at 00400014: 8d060000
lw      $6, 0($8)
New pc = 00400018
Updated r06 to 00000002
No memory location was updated.
Executing instruction at 00400018: 24a50002
addiu   $5, $5, 2
New pc = 0040001c
Updated r05 to 00000004
No memory location was updated.
Executing instruction at 0040001c: 25082000
addiu   $8, $8, 8192
New pc = 00400020
Updated r08 to 00404000
No memory location was updated.
Executing instruction at 00400020: ad050000
sw      $5, 0($8)
Memory Access Exception at 0x00400020: address 0x00404000
```

Output 3:

```
Executing instruction at 00400000: 24040003
addiu   $4, $0, 3
New pc = 00400004
Updated r04 to 00000003
No memory location was updated.
Executing instruction at 00400004: 24050002
addiu   $5, $0, 2
New pc = 00400008
Updated r05 to 00000002
No memory location was updated.
Executing instruction at 00400008: 3c080040
lui     $8, 0x40
New pc = 0040000c
Updated r08 to 00400000
No memory location was updated.
Executing instruction at 0040000c: 25082000
addiu   $8, $8, 8192
New pc = 00400010
Updated r08 to 00402000
No memory location was updated.
Executing instruction at 00400010: ad050000
sw      $5, 0($8)
New pc = 00400014
No register was updated.
Updated memory at address 00402000 to 00000002
Executing instruction at 00400014: 8d060000
lw      $6, 0($8)
New pc = 00400018
Updated r06 to 00000002
No memory location was updated.
Executing instruction at 00400018: 24a50002
addiu   $5, $5, 2
New pc = 0040001c
Updated r05 to 00000004
No memory location was updated.
Executing instruction at 0040001c: 2508e000
addiu   $8, $8, -8192
New pc = 00400020
Updated r08 to 00400000
No memory location was updated.
Executing instruction at 00400020: ad050000
sw      $5, 0($8)
Memory Access Exception at 0x00400020: address 0x00400000
```

Output 4:

```
Executing instruction at 00400000: 24040003
addiu   $4, $0, 3
New pc = 00400004
Updated r04 to 00000003
No memory location was updated.
Executing instruction at 00400004: 24050002
addiu   $5, $0, 2
New pc = 00400008
Updated r05 to 00000002
No memory location was updated.
Executing instruction at 00400008: 3c080040
lui     $8, 0x40
New pc = 0040000c
Updated r08 to 00400000
No memory location was updated.
Executing instruction at 0040000c: 25082000
addiu   $8, $8, 8192
New pc = 00400010
Updated r08 to 00402000
No memory location was updated.
Executing instruction at 00400010: ad050000
sw      $5, 0($8)
New pc = 00400014
No register was updated.
Updated memory at address 00402000 to 00000002
Executing instruction at 00400014: 8d060000
lw      $6, 0($8)
New pc = 00400018
Updated r06 to 00000002
No memory location was updated.
Executing instruction at 00400018: 24a50002
addiu   $5, $5, 2
New pc = 0040001c
Updated r05 to 00000004
No memory location was updated.
Executing instruction at 0040001c: 2108e000
```

Output 5:

omitted due to length, see TEST_CASES *directory for further reference*

Output 6:

```
Executing instruction at 00400000: 24080000
addiu   $8, $0, 0
New pc = 00400004
Updated r08 to 00000000
No memory location was updated.
Executing instruction at 00400004: 2400000a
addiu   $0, $0, 10
New pc = 00400008
No register was updated.
No memory location was updated.
Executing instruction at 00400008: 1408fffe
bne     $0, $8, 0x00400004
New pc = 0040000c
No register was updated.
No memory location was updated.
Executing instruction at 0040000c: 00000000
```

In essence these test cases provided an efficient, but not complete, set of both general and corner cases to test the validity of the simulated instruction interpreter system for MIPS in ***C***. The tests included:

- infinite loops resulting from written $ZERO
- summing first 100 digits
- accessing invalid memory addresses to generate a memory exception
- accessing out−of−range memory to generate a memory exception
- running unsupported instructions
- invoking the ***C*** standard library EXIT(...) calls