Test Plan for Knaves NES

Niko Savas	Joe Crozier	Sam Nalwa
1300247	1311502	1332792

October 23, 2015

Software Engineering 3XA3, Lab #3

Contents

List of Figures

List of Tables

1 Tools for Testing

NESTest[1] - An all inclusive test ROM for NES CPU emulators. The ROM contains test cases for every instruction included in the 6502 instruction set. A link to its source is included in the appendix.

CC65[2] - A compiler to compile 6502 assembly code into machine code to be read by the emulator. A link to the software is included in the references

Visual Studio 2015[3] - The IDE used to code and compile Knaves NES. It includes important debugging tools which will be used in the testing process

2 System Test Cases

These tests will be performed in order as they are co-dependant on each other. For example, Test 2, the SDA Instruction Test must utilize the LDA instruction which is tested in Test 1

2.1 Test 1 - LDA Instruction Test

Type: Dynamic Manual Functional (Black Box) Test

Summary: This test is performed to confirm the accuracy and robustness of the LDA instruction. The LDA instruction loads the passed argument into the A register. The instructions are loaded from a custom ROM, the source code of which will be included in appendix [1].

Process

1. A terminal window is opened in the folder which contains the KnavesNES executable.
2. The command knavesnes ldatest.nes is run.

3. The ldatest.nes ROM is run by the emulator. Upon its completion, a log file is dumped which contains the CPUs final memory state.

4. This state is compared with the desired memory state as outlined in the appendix [2].

Confirmation: The outputted log file is in agreement with the desired memory state outlined in appendix [2].

2.2 Test 2 STA Instruction Test

Type: Dynamic Manual Functional (Black Box) Test

Summary: This test is performed to confirm the accuracy and robustness of the STA instruction. The STA instruction stores the contents of A into the passed memory address. The instructions are loaded from a custom ROM, the source code of which will be included in appendix [3].

Process

1. A terminal window is opened in the folder which contains the KnavesNES executable.

2. The command knavesnes statest.nes is run.

3. The statest.nes ROM is run by the emulator. Upon its completion, a log file is dumped which contains the CPUs final memory state.

4. This state is compared with the desired memory state as outlined in the appendix [4].

Confirmation: The outputted log file is in agreement with the desired memory state outlined in appendix [4].

2.3 Test 3 ADC Instruction Test

Type: Dynamic Manual Functional (Black Box) Test

Summary: This test is performed to confirm the accuracy and robustness of the ADC instruction. The ADC instruction adds the passed value or the contents of the passed memory address to the A register. The instructions are loaded from a custom ROM, the source code of which will be included in appendix [5].

Process

1. A terminal window is opened in the folder which contains the KnavesNES executable.
2. The command knavesnes adctest.nes is run.
3. The adctest.nes ROM is run by the emulator. Upon its completion, a log file is dumped which contains the CPUs final memory state.
4. This state is compared with the desired memory state as outlined in the appendix [6].

Confirmation: The outputted log file is in agreement with the desired memory state outlined in appendix [6].

2.4 Test 4 TAX Instruction Test

Type: Dynamic Manual Functional (Black Box) Test

Summary: This test is performed to confirm the accuracy and robustness of the TAX instruction. The TAX instruction transfers the value in the A register to the X register. The instructions are loaded from a custom ROM, the source code of which will be included in appendix [7].

Process

1. A terminal window is opened in the folder which contains the KnavesNES executable.
2. The command knavesnes taxtest.nes is run.
3. The taxtest.nes ROM is run by the emulator. Upon its completion, a log file is dumped which contains the CPUs final memory state.
4. This state is compared with the desired memory state as outlined in the appendix [8].

Confirmation: The outputted log file is in agreement with the desired memory state outlined in appendix [8].

2.5 Test 5 CMP Instruction Test

Type: Dynamic Manual Functional (Black Box) Test

Summary: This test is performed to confirm the accuracy and robustness of the CMP instruction. The CMP instruction compares the passed argument with the value in register A. If the values are the same, the Z flag bit is set to 1. If A is greater than or equal to the passed value, the C flag bit is set to 1. The instructions are loaded from a custom ROM, the source code of which will be included in appendix [9].

Process

1. A terminal window is opened in the folder which contains the KnavesNES executable.
2. The command knavesnes cmptest.nes is run.
3. The cmptest.nes ROM is run by the emulator. Upon its completion, a log file is dumped which contains the CPUs final memory state.
4. This state is compared with the desired memory state as outlined in the appendix [10].

Confirmation: The outputted log file is in agreement with the desired memory state outlined in appendix [10].

2.6 Test 6 BNE Instruction Test

Type: Dynamic Manual Functional (Black Box) Test

Summary: This test is performed to confirm the accuracy and robustness of the CMP instruction. The BNE instruction checks the value of the Z flag which would have been set by a previous compare operation. If the Z flag is set to 0, the program branches to the passed label. The instructions are loaded from a custom ROM, the source code of which will be included in appendix [11].

Process

1. A terminal window is opened in the folder which contains the KnavesNES executable.
2. The command knavesnes beqtest.nes is run.
3. The beqtest.nes ROM is run by the emulator. Upon its completion, a log file is dumped which contains the CPUs final memory state.
4. This state is compared with the desired memory state as outlined in the appendix [12].

Confirmation: The outputted log file is in agreement with the desired memory state outlined in appendix [12].

2.7 Test 7 Loading a Non-Existent ROM file

Type: Dynamic Manual Functional (Black Box) Test

Summary: This test is used to check the programs ability to detect a

file that does not actually exist, but one that is referenced to while running the emulator. When the user loads a file, that does not actually exist on the computer it is being run from the program should state that No such file exists, and should prompt the user to try again using another file.

Process

1. Open a command window in the directory which contains KnavesNES.
2. Run the command: knavesnes xxyyzz.nes where xxyyzz.nes is a non-existent file
3. The program should terminate and advise the user to recheck the file name and its existence.

2.8 Test 8 Loading a Corrupted ROM file

Type: Dynamic Manual Functional (Black Box) Test

Summary: This test is used to check if the .nes ROM file the user has requested to open is one that is actually a ROM file, as well as one with a full set of instructions to be further processed. If both those criteria are not met, the emulator should prompt the user to try again with a non-corrupted .nes file.

Process

1. Open a command window in the directory which contains KnavesNES.
2. Run the command: knavesnes mario.nes where mario.nes is actually mario.doc with its extension changed.
3. The program should terminate and advise the user to recheck the file as it is corrupted/unreadable.

3 Non-Functional Tests

3.1 Test 1 - CPU Speed

Type: Dynamic Manual Structural (white-box) Test

Summary: This test is executed by checking the frequency and cycles per second of the emulated 6502 CPU. This test will give us a more accurate idea on how close our cpu emulation is to the actual 6502 CPU, and is one that must be executed as an emulator relies heavily on the CPU being able to keep up so that there is no delay within the system causing lag later on when the graphics need to be rendered. This test is also required to make sure that the emulated process is not running faster than required, since that would also have a negative impact once the entire emulator is constructed including the graphics rendering module.

Process: This test will be done within the code by using the inbuilt system timer. The time taken by each instruction set will be measured, and then the frequency and cycles per second of the processor will be calculated. The test will then be repeated over and over again until the desired emulated processor speed is achieved.

3.2 Test 2 - Usability Test

Type: Dynamic Manual Functional (white-box) Test

Summary: This test is done to ensure that the software is easy to use. The test requires at least one test subject who is presently unfamiliar with the software, but is capable of basic BASH commands.

Process: The test subject is instructed to enter the command knavesnes –help which echo a short help document into the console. The test subject is then instructed to use this information to successfully run a ROM file in the current directory. The test is deemed successful if the subject is able to execute the ROM without any further assistance.

4 Automated Testing

Automated testing can be done using a bash script, which runs through a list of .nes ROM files, and for each file runs knavesnes, takes a screenshot of the screen, and goes onto the next one. These screenshots can be quickly scrolled through to make sure all the games have executed properly. This list can be as big as a 1000 games, and the test can be automated to run over and over again to make sure it always produces the correct results.

5 Proof of Concept/Existing Implementation Test

For our proof of concept, we will stress test a previously made emulator that we have already compiled using the NESTest rom. By running this previously created emulator with the testing ROM, it will give us a better understanding of the robustness of the emulator and a better idea as to how our system should respond to the testing module.

6 Schedule of Testing

Date of Test	What is Tested	Group Members doing Testing
30th October, 2015	Y.A.N.E. is tested using NESTest to get an idea of robustness	All Members
5th November, 2015	CPU Instructions outlined in the testing document are tested.	Joe
10th November, 2015	knavesnes ROM reading is tested using NESTest to see if all the instructions are readable	Sam
15th November, 2015	knavesnes CPU emulator is tested to ensure that all 6502 instructions work correctly.	Niko
25th November, 2015	knavesnes is tested from start to finish using a CPU test ROM file.	All Members

Table 1: Schedule of Testing for Knaves NES

References

[1] *NESTest ROM* - http://nickmass.com/images/nestest.nes

[2] *CC65* - http://www.cc65.org/

[3] *Visual Studio 2015* - https://www.visualstudio.com/en-us/products/vs-2015-product-editions.aspx

Appendices

Appendix 1

```
LDA #$c0
```

Appendix 2

```
A = $c0
X = $00
Y = $00

All mem = $00
```

Appendix 3

```
LDA #$10
STA $01
LDA #$20
STA $02
LDA #$c0
STA $c2
LDA #$fe
STA $00
```

```
LDA #$19
STA $00
```

Appendix 4

```
A = $19 X = $00 Y = $00

NV-BDIZC
00110000
```

Appendix 5

```
LDA #$10
STA $01
ADC $01
LDA #$01
STA $01
ADC $12

LDA #$03
STA $02
ADC $23
LDA #$05
STA $03
ADC $34
LDA #$ff
STA $03
ADC $34
```

Appendix 6

```
A=$c2 X=$00 Y=$00

NV-BDIZC
10110000
```

```
0000: 00 01 03 c2 00 00 00 00 00 00 00 00 00 00 00 00
0010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
...
```

Appendix 7

```
LDA #$01
TAX
LDA #$ff
TAX
LDA #$50
TAX
LDA #$30
TAX
LDA #$21
TAX
```

Appendix 8

```
A=$21 X=$21 Y=$00
NV-BDIZC
```

```
00110000

All mem = $00
```

Appendix 9

```
DA #$13
STA #$00
CMP #$13
```

Appendix 10

```
A=$13 X=$00 Y=$00
NV-BDIZC
00110011

0000: 13 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Appendix 11

```
  LDA #$02
```

```
  STA $01
add:
  ADC $01
  CMP #$06
  BNE add
  LDA #$32
  BRK
```

Appendix 12

```
A=$32 X=$00 Y=$00
NV-BDIZC
00110001

0000: 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Revision History

Revision	Date	Author(s)	Description
1.0	Oct 23, 2015	Niko Savas	created