CS 219: Homework #3

Due on September 21, 2016 at 4:00pm

Dr. Egbert

Matthew J. Berger

Problem 1

1.2.) A.) On the IAS, what would the machine code instruction look like to load the contents of memory address 2 to the accumulator?

Answer: (with spaces between bytes for readability)

Opcode	Operand
0000 0001	0000 0000 0010

B.) How many trips to memory does the CPU need to make to complete this instruction during the instruction cycle?

Answer:

Initially, the CPU has to fetch the instruction at the program counter in memory. The instruction contains the address of the data to load. When executed, the memory is accessed in order to load the actual data located at that address in memory. This requires two trips to memory during the instruction cycle.

1.3.) On the IAS, describe in English the process that the CPU must undertake to read a value from memory and to write a value to memory in terms of what is put into the MAR, MBR, address bus, data bus, and control bus.

Answer:

- To read a value from memory, the CPU places the address from the opcode in the MAR. Then the CPU will send a signal on the read control line to memory and puts the address on the address bus. After this, the memory will copy the contents of the memory location that was relayed by the data bus. Finally, the data is transferred to the MBR.

- To write a value to memory, the CPU places the address from the opcode in the MAR. Then the cpu puts the data into the MBR. After, the CPU sends a signal on the write control line to memory and puts the address on the address bus, as well as the data on the data bus. Finally, the memory transfers the data on the data bus into the memory location relayed by the address bus.

Problem 1 continued on next page. . .

1.4.) Given the memory contents of the IAS computer shown below,

Address	Contents
08A	010FA210FB
08B	010FA0F08D
08C	020FA210FB

Show the assembly language code for the program, starting at address 08A. Explain what this program does.

Answer:

Address	Instruction	Argument
08A	LOAD	M(0FA)
	STOR	M(0FB)
08B	LOAD	M(0FA)
	JUMP	+M(08D)
08C	LOAD	-M(0FA)
	STOR	M(0FB)

This program stores the absolute value of the data at memory location 0x0FA into memory location 0x0FB.

1.5.) In Figure 1.6, indicate the width, in bits, of each data path (e.g., between AC and ALU).

Answer:

- Data paths to and from the MAR total to 12 bits.
- Data paths to and from the MBR total to 40 bits.
- All paths to and from the AC is 40 bits.
- All paths to and from the MQ is 40 bits.

Problem 2

2.2) Consider two different machines, with two different instruction sets, both of which have a clock rate of 200 MHz. The following measurements are recorded on the two machines running a given set of benchmark programs:

Instruction Type	Instruction Count (millions)	Cycles Per Instruction
Machine A		
Arithmetic and Logic	8	1
Load and Store	4	2
Branch	2	4
Others	4	3
Machine B		
Arithmetic and Logic	10	1
Load and Store	8	2
Branch	2	4
Others	4	3

A.) Determine the effective CPI, MIPS rate, and execution time for each machine.

B.) Comment on the results.

Answer:

Machine A:

- $CPI_A = \frac{\sum CPI_i \times I_i}{I_c} = \frac{((8 \times 1) + (4 \times 3) + (2 \times 4) + (4 \times 3)) \times 10^6}{(8 + 4 + 2 + 4) \times 10^6} \approx 2.22$
- $MIPS_A = \frac{f}{CPI_A \times 10^6} = \frac{200 \times 10^6}{2.22 \times 10^6} = 90$
- $CPU_A = \frac{I_c \times CPI_A}{f} = \frac{18 \times 10^6 \times 2.22}{200 \times 10^6} = 0.20s$

Machine B:

- $CPI_B = \frac{\sum CPI_i \times I_i}{I_c} = \frac{((10 \times 1) + (8 \times 2) + (2 \times 4) + (4 \times 3)) \times 10^6}{(10 + 8 + 2 + 4) \times 10^6} \approx 1.92$
- $MIPS_B = \frac{f}{CPI_B \times 10^6} = \frac{200 \times 10^6}{1.92 \times 10^6} = 104$
- $CPU_B = \frac{I_c \times CPI_B}{f} = \frac{24 \times 10^6 \times 1.92}{200 \times 10^6} = 0.23s$

Machine B has a higher MIPS than Machine A, but it takes a slightly longer CPU time to execute the same set of benchmark instructions and is therefore slightly less efficient than Machine A.

Problem 2 continued on next page...

2.4) Four benchmark programs are executed on three computers with the following results:

	Computer A	Computer B	Computer C
Program 1	1	10	20
Program 2	1000	100	20
Program 3	500	1000	50
Program 4	100	800	100

The table shows the execution time in seconds, with 100,000,000 instructions executed in each of the four programs. Calculate the MIPS values for each computer for each program. Then calculate the arithmetic and harmonic means assuming equal weights of the four programs, and rank the computers based on arithmetic and harmonic mean.

$$\text{MIPS rate is } MIPS = \frac{I_c}{T \times 10^6}$$

$$\text{Arithmetic mean is } R_A = \frac{1}{M}\sum_{i=1}^{m} R_i$$

$$\text{Harmonic mean is } R_H = \frac{m}{\sum_{i=1}^{m}\frac{1}{R_i}}$$

With the MIPS formula:

$MIPS = \frac{I_c}{T \times 10^6} = \frac{100{,}000{,}000}{T \times 10^6} = \frac{100}{T}$

Thus, the MIPS values are as follows:

	Computer A	Computer B	Computer C
Program 1	100	10	5
Program 2	0.1	1	5
Program 3	0.2	0.1	2
Program 4	2	0.125	1

	Arithmetic Mean	Rank
Computer A	25.575	1
Computer C	3.25	2
Computer B	2.80	3

	Harmonic Mean	Rank
Computer C	2.1	1
Computer A	0.25	2
Computer B	0.21	3

Problem 3

Download the Simulated Computer zip and pdf files from the Simulated Computer Folder on Canvas. Unzip and install the executable file on your own computer. The provided version actually runs on an ATARI 800 computer. An emulator for the ATARI 800 is included in the zip file. The program to execute after un-zipping is 'ATARI800Win.exe'. The first time you run the emulator you should click on File → Autoboot Image → Simulated Computer II.atr. After this, when you execute the emulator the Simulated Computer program should automatically start. The provided PDF is the user's manual for the simulated computer.

After you review Appendices II, III, and IV of the user's manual, write assembly language programs for the following three exercises. Include screen shots and separate program listings for exercises.

Program listings should have one line per instruction/address as follows:

Line Number	Instruction	Comments
03	LDA15	// Load the value 15 into the accumulator

Problem 3 continued on next page...

Evaluating Algebraic Expressions

3.1) Those of you who have stuided algebra may recall that you were occasionally required to calculate the value of an expression when particular values are substituted for variables. For example, the expression $7X$ is equal to 14 if $X = 2$. The value is 42 if $X = 6$. The value of the expression $3X - 5$ is 13 if you substitute 6 for X. (Note: 3X means 3 times X.)

Write a computer program which will evaluate the expression: $7X - 8$. Check your program by running it and inputting the numbers given below.

Input	Output
3	13
43	293

Solution:

Line Number	Instruction	Comments
00	INP14	// Input a value to location 14
01	LDA14	// Load the value at location 14 into the Accumulator
02	MUL12	// Multiply the value in the Accumulator by the value at location 12
03	SUB13	// Substract the value in the Accumulator by the value at location 13
04	STA14	// Store the value in the Accumulator at location 14
05	OUT14	// Output the value at location 14
06	STP	// HALT

A screenshot of this program can be found on the next page:


Atari800Win PLus 3.1: XL/XE (64 KB)
File Atari Input View Sound Misc Help
RUN
14:43
DIGO-6
ALPO-6
CPU
EXEC INC PC FETCH
AC 293
PC 07
IR 000
INVALID COMMAND
13
293
MEMORY
00 INP14
01 LDA14
02 MUL12
03 SUB13
04 STA14
05 OUT14
06 STP
07
08
09
10
11
12 007
13 008
14 293
15
16
17
18
19
20
21
22
23
S
C
T
D
Speed: 100%

Number Sequences

3.2) A number sequence is a series of numbers that follow a particular "rule", or pattern, as they go from one to the next. For example, the sequence:

(a) 3, 6, 12, 24, 48, 96, etc.

uses the rule "multiply by 2". Can you guess the rule for the following sequence?

(b) 2, 5, 11, 23, 47, 95, etc.

(It is "times 2 then add 1".)

Once you know the rule for a sequence, you can write a program which will output that sequence. First, write programs that output sequences (a) and (b).

Then write a program that outputs the Fibonacci sequence:

(c) 1, 1, 2, 3, 5, 8, 13, 21, 34, etc.

Solution: Sequence A

Line Number	Instruction	Comments
00	LDA14	// Load the loop counter's current value - Initialized to 6 in memory
01	SUB15	// Subtract one from the loop counter
02	STA14	// Store the decremented loop counter at the same address it was loaded from
03	LDA12	// Load the current value in the sequence
04	OUT12	// Output the current value
05	MUL13	// Multiply the current value by 2 - Address 13 is preinitialized to 2 in memory
06	STA12	// Store the current value, overwriting the original location
07	LDA14	// Load the loop counter's current value into the accumulator
08	SKP4	// Skip the next instruction if the value in the Accumulator is less than or equal to 0
09	JMP1	// Jump to address 01
10	STP	// HALT

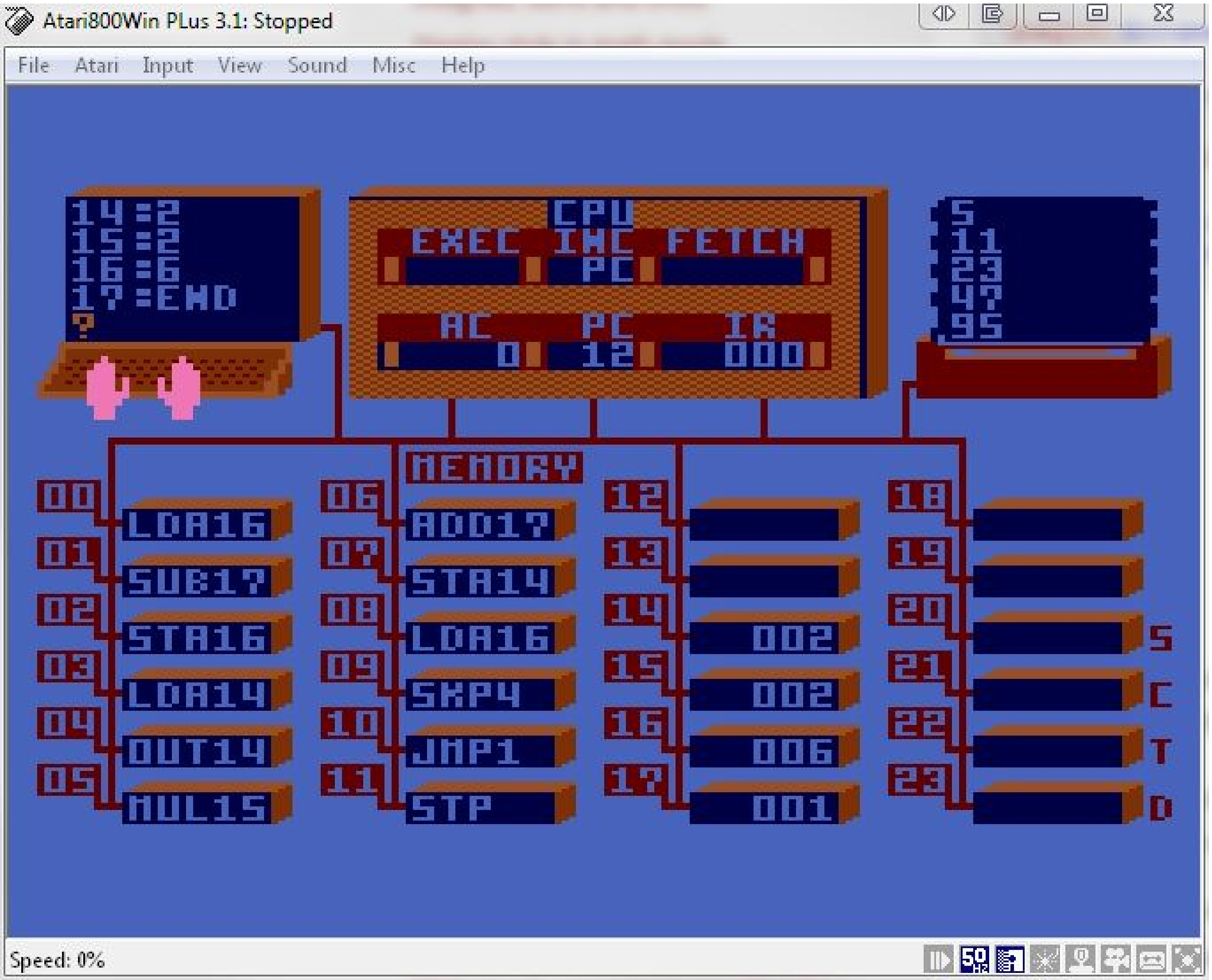
A screenshot of this program can be found on the next page:


Atari800Win PLus 3.1: Stopped
File Atari Input View Sound Misc Help
14 :6
15 :1
16 :END
ALPO-11
CPU
EXEC INC FETCH
PC
AC PC IR
0 11 000
6
12
24
48
96
MEMORY
00 LDA14
01 SUB15
02 STA14
03 LDA12
04 OUT12
05 MUL13
06 STA12
07 LDA14
08 SKP4
09 JMP1
10 STP
11
12 003
13 002
14 006
15 001
16
17
18
19
20
21
22
23
S
C
T
D
Speed: 0%

Solution: Sequence B

Line Number	Instruction	Comments
00	LDA16	// Load the loop counter's current value - Initialized to 6 in memory
01	SUB17	// Subtract one from the loop counter
02	STA16	// Store the decremented loop counter at the same address it was loaded from
03	LDA14	// Load the current value in the sequence
04	OUT14	// Output the current value to the screen
05	MUL15	// Multiply the current value by 2 - Address 15 is preinitialized to 2 in memory
06	ADD17	// Add 1 to the current value
07	STA14	// Store the current value, overwriting the original location
08	LDA16	// Load the loop counter's current value into the accumulator
09	SKP4	// Skip the next instruction if the value in the Accumulator is less than or equal to 0
10	JMP1	// Jump to address 01
11	STP	// HALT

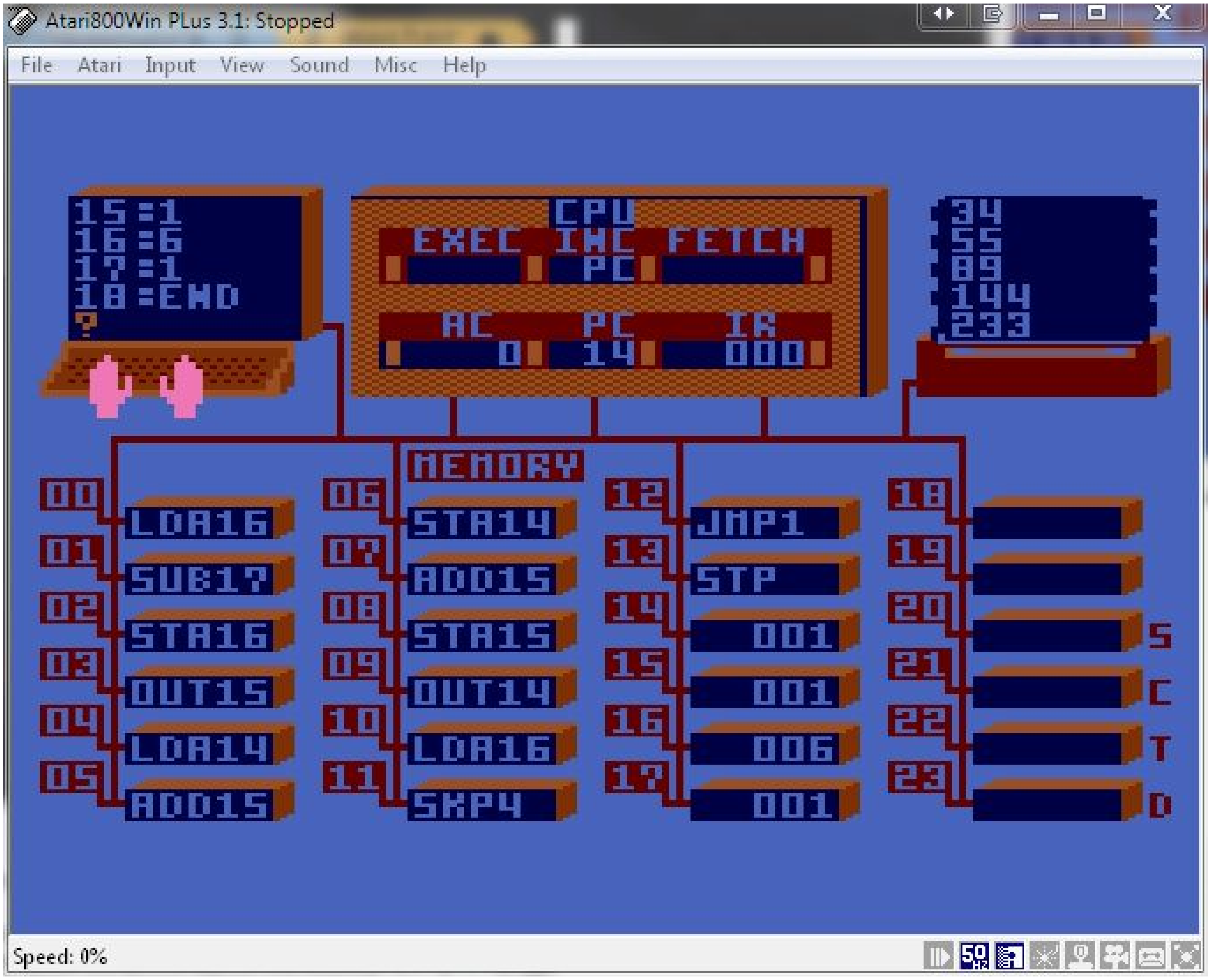
A screenshot of this program can be found on the next page:


Atari800Win PLus 3.1: Stopped
File Atari Input View Sound Misc Help
14 =2
15 =2
16 =6
17 =END
?
CPU
EXEC INC PC FETCH
AC PC IR
0 12 000
5
11
23
47
95
MEMORY
00 LDA16
01 SUB17
02 STA16
03 LDA14
04 OUT14
05 MUL15
06 ADD17
07 STA14
08 LDA16
09 SKP4
10 JMP1
11 STP
12
13
14 002
15 002
16 006
17 001
18
19
20
21
22
23
S
C
T
D
Speed: 0%

Solution: Sequence C

Line Number	Instruction	Comments
00	LDA16	// Load the loop counter's current value - Initialized to 6 in memory
01	SUB17	// Subtract one from the loop counter
02	STA16	// Store the decremented loop counter at the same address it was loaded from
03	OUT15	// Output current value in the sequence
04	LDA14	// Load the next sequence value from location 14 into the Accumulator
05	ADD15	// Add the prior sequence value to the Accumulator
06	STA14	// Store the current value at the 'next value' address
07	ADD15	// Add the previous value to the Accumulator value to get the next value in the sequence
08	STA15	// Store the Accumulated value, this becomes the 'previous value' in the next iteration
09	OUT14	// Output the current value. Both previous and next value are output each iteration.
10	LDA16	// Load the loop counter's current value
11	SKP4	// Skip the next instruction if the value in the Accumulator is less than or equal to 0
12	JMP1	// Jump to address 01
13	STP	// HALT

A screenshot of this program can be found on the next page:


Atari800Win PLus 3.1: Stopped
File Atari Input View Sound Misc Help
15 =1
16 =6
17 =1
18 =END
?
CPU
EXEC INC FETCH
PC
AC PC IR
0 14 000
34
55
89
144
233
MEMORY
00 LDA16
01 SUB17
02 STA16
03 OUT15
04 LDA14
05 ADD15
06 STA14
07 ADD15
08 STA15
09 OUT14
10 LDA16
11 SKP4
12 JMP1
13 STP
14 001
15 001
16 006
17 001
18
19
20
21
22
23
S
C
T
D
Speed: 0%

A Decision Maker

3.3) Write a program that uses an INP instruction. If the number that you INPut is positive or zero, have your program OUTput the number 1. If the number that you INPut is negative, have the program OUTput the number zero.

Solution: Sequence C

Line Number	Instruction	Comments
00	INP14	// Input a number to location 14
01	LDA14	// Load the value at location 14 into the Accumulator
02	SKP5	// Skip the next instruction if the value in the Accumulator is positive
03	OUT13	// Output the value stored at location 13 - Initialized to 0
04	SKP1	// Skip the next instruction if the value in the Accumulator is negative
05	OUT12	// Output the value stored at location 12 - Initialized to 1
06	STP	// HALT

A screenshot of this program can be found on the next page:


Atari800Win PLus 3.1: Stopped
File Atari Input View Sound Misc Help
07:END
RUN
14:-10
ALPO-10
CPU
EXEC INC PC FETCH
AC -10
PC 07
IR 000
1
0
1
0
MEMORY
00 INP14
01 LDA14
02 SKP5
03 OUT13
04 SKP1
05 OUT12
06 STP
07
08
09
10
11
12 001
13 000
14 -010
15
16
17
18
19
20
21
22
23
S
C
T
D
Speed: 0%