Lab 4:

Project 4:: Computer Architecture Lab

By. Melvin Evans
(ID#4692)

CSC 137
Professor Daryl Posnett
May 10, 2021

Problem Number One

8.2 Consider the Acc-ISA assembly instructions "LD data" (ACC ← data), "LD (adrs)" (ACC ← Memory[adrs]), "ST (adrs)" (Memory[adrs] ← ACC), "ADD (adrs)" (ACC ← ACC + Memory[adrs]), "XOR (adrs)" (ACC ← ACC M[adrs]). Do the following:

a. Write an assembly program for the following program:

```
X = -2;
Y = 6;
Z = 11;
T = X + Y - Z;
```

b. For the assembly instructions, draw a single-cycle instruction data path.

Solution:

A.

Xor

a	b	R
0	0	0
0	1	1
1	0	1
1	1	0

A Xor 1 is an inverter. Allowing us to represent negative numbers in binary.

Memory locations being used:

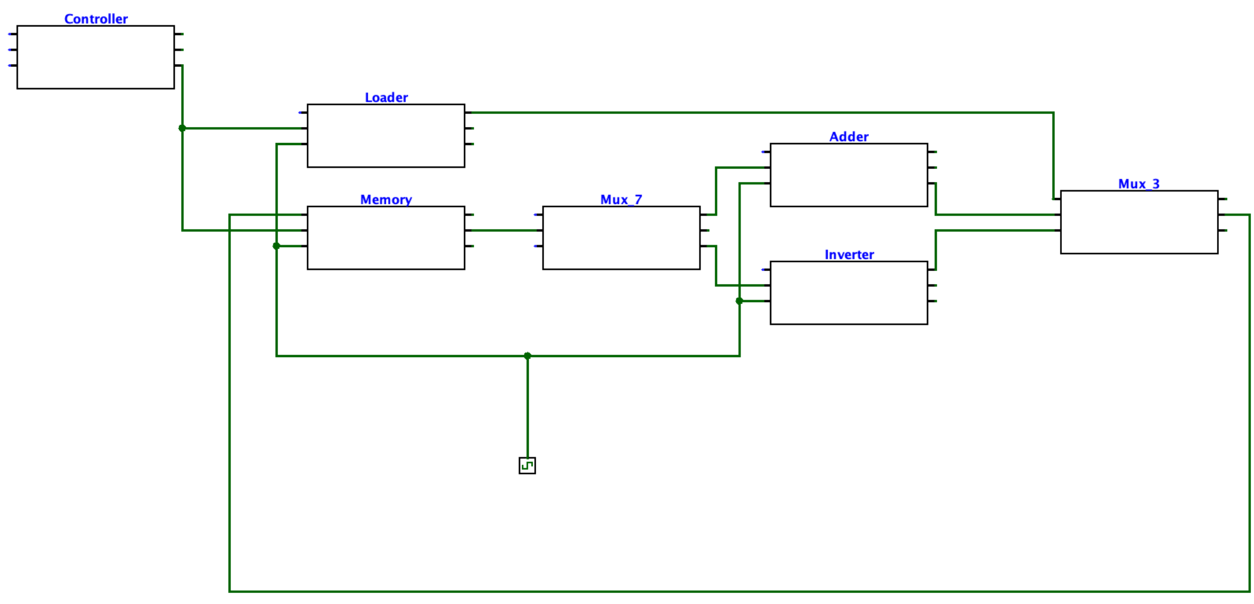
0	1	2	3	4	5	6	7
Inv bits	x	y	z	t	Temp var	result	Number 1 for converting from 1's comp to 2's

Instructions:

Instruction	binary	Description of instruction
Ld(data)	00001	Loads 1 into accumulator

Str	00111	Stores 1 into memory location 7
Ld(data)	11111	Loads all ones to be used as an inverter into accumulator
Str	00000	Stores all one into memory location 0
Ld(data)	00010	Loads 2 into accumulator
Xor	00000	Bitwise inverts 2
Add	00111	Adds 1 to get -2
Str	00001	Stores -2 at memory location 1
Ld(data)	00110	Loads 6 into accumulator
Str	00010	Stores 6 at memory location 2
Ld(data)	01101	Loads 11 into the accumulator
Str	00011	Stores 11 at memory location 3
Ld(addrs)	00001	Loads memory location 2 into accumulator
Add	00010	Add -2 and 6
Str	00101	Sore 4 into memory location 5
Ld(addrs)	00011	Loads memory location 3 into accumulator
Xor	00000	Bitwise inverts 11
Add	00111	Adds 1 to get -11
Add	00101	Adds 4 and -11
Str	00110	Stores -7 at memory location 6 (result)

B.



Problem Number Two

8.3 An Acc-ISA CPU executes the following instructions using 3-bit op-codes and 5-bit address or 2’s complement data. Do the following:

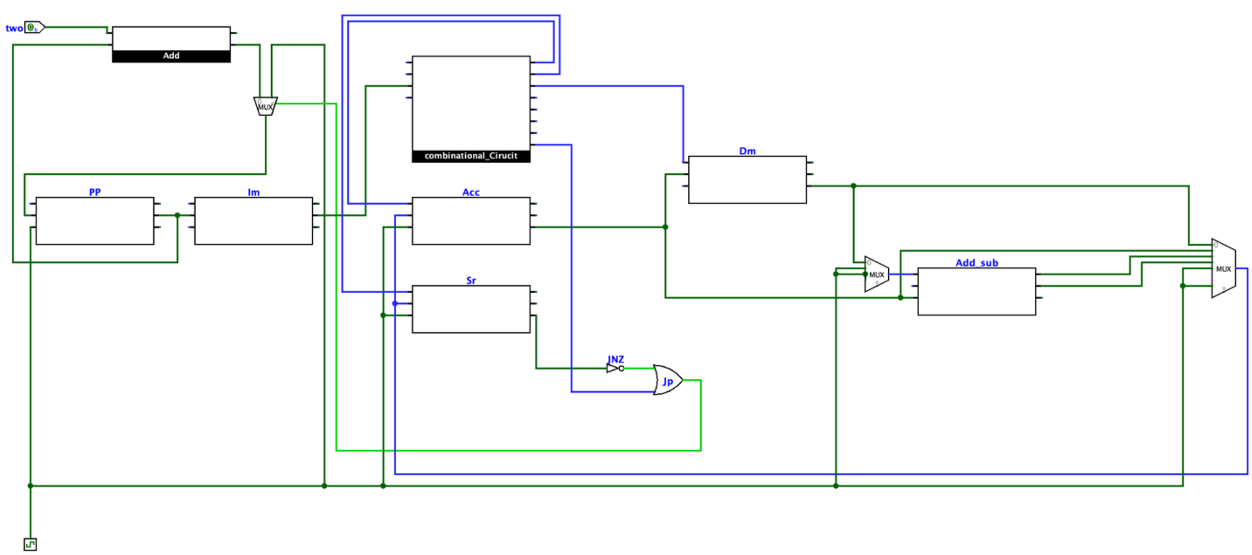
```
LD (address)      //Acc←Memory [address], read from LM2
LD data           //Acc←data (a 2's complement number, sign
                  //extended)
ADD data          //Acc←Acc + data (data is a 2's complement
                  //number, sign extended)
SUB data          //Acc←Acc - data (data is a 2's complement
                  //number, sign extended)
ADD (address)     //Acc←Acc + Memory[address]
SUB (address)     //Acc←Acc - Memory[address]
ST (address)      //M[address]←Acc
JMP address       //PP←address
JZ address        //PP←address if ACC = 0
```

a. Draw a data path for the CPU, assuming the DM has separate input and output buses, as in the data path shown in Fig. 8.7. Do not include data paths not used by the instructions.

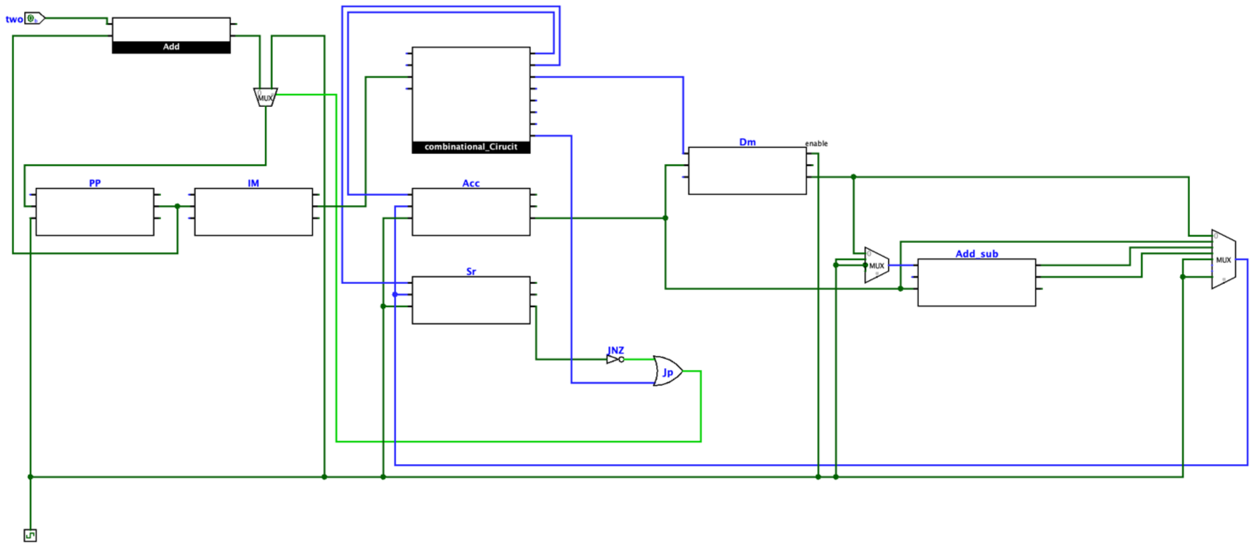
b. Draw a data path for the CPU, assuming the DM has a bidirectional data bus. Do not include data paths not used by the instructions.

Solution:

A.



B.



Problem Number Three

8.4 For the high-level code segment shown next, create a set of instructions and write an equivalent assembly program for each of the following architectures:

```
int i, sum;
for (i=0; i> k, i++)
   if (i mod 2 == 0)
         sum = sum +i;
```

a. Stack-ISA

b. Acc-ISA

c. CISC-ISA

d. RISC-ISA

A.

Commands:

Command	What it does
Push	Push onto stack
Pop	Pop off of stack

Add	Addition
Inc	Add one to data
Mod	Modulous
Str	Store
Jmp	Jump
JNE	Jump if Not equal
JGt	Jump if greater than
JE	Jump if equal

Instructions:

Instruction	Attribute 1	Attribute 2	Attribute 3
Jmp	Result		
Stop:			
Jmp	Top		
Add	(sum)	(i)	
JNE	Top		
Mod	(i)	2	
JGT	(k)	Stop	
Inc	(i)		
Top:			
Pop			
JE	#0	(sum)	Top
Result:			
Push	(i)		
Push	(sum)		
Str	#0	(sum)	
Str	#0	(i)	

B.

Commands:

Mod	%
JLT	Jump if <
JNZ	Jump if !=

COMPARE	Compares 2 numbers

Instructions:

Instruction	Info being used
LOAD	0
ST	0
LOAD	0
ST	1
LOAD	1
L1:	
COMPARE	0, (K)
JLT	STOP
MOD	0, (2)
COMPARE	ACC
JNZ	L1
LOAD	1
ADD	0
ST	1
LOAD	0
ADD	2
ST	0
JMP	L1
STOP:	
DO NOTHING	

C.

Instruction	Address 1	Address 2
Ld	R3	#2
Ld	R1	#0
Ld	R2	#0
L1:		
Compare	R1	K

JLT	Stop	
Ld	R4	R1
Mod	R4	R3
Compare	R4	#0
JE	L1	
Add	R2	R1
JMP	L1	
Stop:		
Nop		

D.

Instruction	Address 1	Address 2
Ld	R1	#0
Ld	R2	#0
L1:	R1	
Compare	Stop	K
JLT	R4 R1	
Mod	R4	#2
Compare	R4	#0
JE	L1	
Add	R2 R2	R1
JMP	L1	
Stop:		
Nop		

All commands are added to the list when they are introduced previously stated commands are not repeated.

Problem Number Four

10.1. We would like to improve the estimated average memory latency in Example 10.1. Suppose, instead of SDRAMs, the memory unit is designed using DDR SDRAMs. Recalculate the average memory latency.

Solution:

Original equation with SRAM:

Avg latency = (.95)(1ns)+(1-.95)(.90)(3ns)+(1-.95)(1-.90)(20ns) = 1.185ns

Equation with DDR SRAM:
Avg latency = (.95)(1ns)+(1-.95)(.90)(2ns)+(1-.95)(1-.90)(20ns) = 1.05ns

Problem Number Five

10.3. Consider the following four memory locations accessed *N* times in a loop by CPU, and suppose a memory address is partitioned into tag, slot, and offset, as shown. Do the following:

0x3C1C (16-bit address)

0x0421
0x041F
0x0C88

Tag	Slot	Offset
6	6	4

a. Determine the number of misses in the first round, assuming the cache is initially empty.

b. Determine the total number of misses for *N* rounds, assuming the cache is initially empty.

c. Suppose the cache mapping is changed to a two-way set associative. Calculate the tag, set, and offset field sizes, and then determine the number of misses in the first round, assuming the cache is initially empty.

d. Determine the total number of misses for *N* rounds, assuming the cache is initially empty and a cyclic replacement policy (if needed) is used.

Solution:

A.

	Tag	Slot	Offset
0x3c1c	001111	000001	1100
0x0421	000001	000010	0001
0x041f	000001	000001	1111

0x0c88	000011	001000	1000

Round 1:

Cold Miss
Cold Miss
Conflict Miss
Cold Miss
Total = 4 misses [3 cold miss + 1 conflict miss]

B.

Round 2:

Conflict Miss
Hit
Conflict Miss
Hit

Round n:

Total = 4 + (N-1) * 2 [conflict misses/round]

C.

	Tag	Set	Offset
0x3c1c	0011110000	01	1100
0x0421	0000010000	10	0001
0x041f	0000010000	01	1111
0x0c88	0000110010	00	1000

Offset = $\text{Log}_2 16 = 4$
Set = 2 (two set associative)
Tag = 16 – 4 – 2 = 10

Round 1:

Cold Miss slot 0
Cold Miss slot 0
Cold Miss slot 1
Cold miss slot 0

Total = 4 (4 Cold Misses)

D.

Round n:

Hit
Hit
Hit
Hit

Total = 4 (4 Cold Misses)

Problem Number Six

10.4. Repeat Exercise 10.3(a) through (d) for the following four addresses:
0x0C1C (16-bit address)
0x0521
0x041F

0x4D28

Solution:

A.

	Tag	Slot	Offset
0x0c1c	000011	000001	1100
0x0521	000010	100010	0001
0x041f	000001	000001	1111
0x4d28	010011	010010	1000

Round 1:

Cold Miss
Cold Miss
Conflict Miss
Cold Miss
Total = 4 misses [3 cold miss + 1 conflict miss]

B.

Round 2:

Conflict Miss
Hit
Conflict Miss
Hit

Round n:

Total = 4 + (N-1) * 2 [conflict misses/round]

C.

	Tag	Set	Offset
0x0c1c	0000110000	01	1100
0x0521	0000101000	10	0001
0x041f	0000010000	01	1111
0x4d28	0100110100	10	1000

Offset = $\text{Log}_2 16 = 4$
Set = 2 (two set associative)
Tag = 16 – 4 – 2 = 10

Round 1:

Cold Miss slot 0
Cold Miss slot 0
Cold Miss slot 1
Cold miss slot 1

Total = 4 (4 Cold Misses)

D.

Round n:

Hit
Hit
Hit
Hit

Total = 4 (4 Cold Misses)

Problem Number Seven

10.5. Repeat Exercise 10.3(a) through (d) for the following four addresses:

0x3C1F (16-bit address)

0x042C
0x0460
0x3C1D

Solution:

A.

	Tag	Slot	Offset
0x3c1f	001111	000001	1111
0x042c	000001	000010	1100
0x0460	000001	000110	0000
0x3c1d	001111	000001	1101

Round 1:

Cold Miss
Cold Miss
Cold Miss
Hit

Total = 3 misses [3 cold mises]

B.

Round 2:

Hit
Hit
Hit
Hit

Round n:

Total = 3 misses [3 cold misses]

C.

	Tag	Set	Offset
0x3c1f	0011110000	01	1111
0x042c	0000010000	10	1100
0x0460	0000010001	10	0000
0x3c1d	0011110000	01	1101

Offset = $\text{Log}_2 16 = 4$
Set = 2 (two set associative)
Tag = 16 – 4 – 2 = 10

Round 1:

Cold Miss slot 0
Cold Miss slot 0
Cold Miss slot 1

Hit

Total = 3 [3 Cold Misses]

D.

Round n:

Hit
Hit
Hit
Hit

Total = 3 [3 Cold Misses]

Problem Number Eight

10.17. Suppose a system has 16 KB virtual memory space, 16 B page size, and 2 KB physical memory. Do the following:

a. Determine the number of virtual and physical pages.

b. Assuming that each page table entry is 2 B, what is the maximum size of a page table?

c. Design a page table organization to translate a 16-bit virtual address to an 11-bit physical address.

Solution:

A.

Virtual: 16kb/16b = 1024 pages
Physical: 2kb/16b = 128 pages

B.

16b/2b = 8 bit

C.

16 bit virtual address has a 7 bit offset for physical addresses. This is found by doing Log_2(amount of physical pages) which in this case is $\text{Log}_2(128)$. This gets us 7 which represents the offset of physical address.

The 9 upper bits of the 16 bit virtual address are used to access the physical memory location that is found inside of the main memory. While the lower 7 are sent directly down to the offset of the physical memory address. Together they form the 11 bit physical location address needed to map data to a physical memory location.

Problem Number Nine

10.18. Consider a TLB; answer the following questions:

a. Briefly explain the purpose for a TLB (e.g., what if no TLB is used?).

b. Explain why a TLB should be designed as fully associative cache (e.g., what if it is implemented as a direct-mapped cache?).

Solution:

A.

A translation lookaside buffer is used to decrease the amount of time it takes to look up the physical memory location that is found in main memory. Without it the process has long latency increasing the amount of time inquiries take.

B.

A TLB is setup as a fully associative cache so that the retrieved data can be put inside any unused memory location. Allowing less conflicts than if we used direct mapping which would have us be forced to specify slot#. Along with that we do not use 2 set associative due to it not always replacing the last used location.