Computer Architecture EECE7352 Homework 3

NUID: 001055828 NAME:MINGZHE ZHANG :

Part A:

Provide the details of the microarchitecture of the pipeline

processor/characteristics	ARM9TDMI[1]	Intel 8051 MCS-251[2] [3]
number of pipeline stages	5(fetch,decode,execute,memory access and write-back)	3(fetch decode execute)
number of instructions issued per cycles	1.1 MIPS/MHz	1MIPS/12MHz
CPI	1.5	12
Typical Max Clock rate	120 MHz	100 MHz

reference:[1]The ARM9Family-HIgh Performance Microprocessors for Embedded Applications, Simon Segars, Manager CPU Development, ARM Ltd.

[2]john Wharton: An Introduction to the Intel MCS-51 Single-Chip Microcomputer Family, Application Note AP-69, May 1980, Intel Corporation.

[3]"MCS 51, MCS 251 and MCS 96 Microcontroller Product Lines, the Intel 186, Intel386 and Intel486 Processors Product Lines, and the i960 32 Bit RISC Processor, PCN 106013-01, Product Discontinuance, Reason for Revision: Add Key Milestone information and revise description of change"

Part B

Write a simple simulator that models the 2 predictors and processes the instruction trace provided. Submit both the code for your branch predictor simulator (on Canvas) and report on the number of buffer misses (first time taken branches), and the number of correct and

incorrect predictions for this address trace for each predictor. Add a discussion explaining why one predictor does better than the other.

```
C:\Users\Administrator\Desktop\google云文件夹\computer architecture\module3\HWK3>predict.out
buff misses:109895
1-bit predictor:80.5419%
correct: 873603 uncorrect: 211055
2-bit predictor:89.4567%
correct: 970299 uncorrect: 114359

C:\Users\Administrator\Desktop\google云文件夹\computer architecture\module3\HWK3>
```

The 2-bits predictor is better than 1-bit one. Because it will reduce the mistakes when many not taken scenarios are detected and taken scenarios is in it or not taken scenarios in many taken scenarios. There will be 2 status of weakly one for infrequent reverse scenarios.

As an added step, experiment with other prediction algorithms (besides a 2-bit counter). You only have 8 entries in total for these alternative algorithms. The best performing algorithm will receive 25 extra credit points

```
C:\Users\Administrator\Desktop\google云文件夹\computer architecture\module3\HWK3>g++ mypredict.cpp -o mypredict.out

C:\Users\Administrator\Desktop\google云文件夹\computer architecture\module3\HWK3>mypredict.out
buff misses:109895
my predictor:89.455%
correct: 970281 uncorrect: 114377

C:\Users\Administrator\Desktop\google云文件夹\computer architecture\module3\HWK3>
```

```
    set<long long> jumped{};

    bool prediction=false;
    for(int i=1;i<address.size();i++){
        if(prediction==jump[i-1])
            cnt++;
        if(jumped.count(address[i])){
            prediction=true;
        }else
            prediction=false;
        if(jump[i-1])
            jumped.insert(address[i-1]);
    }
```

My algorithm is about recording address that has been taken "jump" action. And if the current address is in the set "jumped", it will predict "taken" or it will predict "not taken". Every time it is jumped, it will record last address to the set.

Part C:

C.1

a. There are dependencies for register x1 from ld to the addi, register x1 from adii to the sd, register x2 from addi to sub and register x4 from sub to bnez.

b. As refferred in section a, first addi should wait ld, sd should wait first addi, second addi should wait sd and bnez should wait sub. Also, this is a loop, so there will be ld waiting for bnez at the end of the loop. So the progress is like this.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	ld	IF	ID	EX	MEM	WB													
3	addi		IF	STALL	STALL	ID	EX	MEM	WB										
4	sd					IF	STALL	STALL	ID	EX	MEM	WB							
5	addi								IF	ID	EX	MEM	WB						
6	sub									IF	STALL	STALL	ID	EX	MEM	WB			
7	bnez												IF	STALL	STALL	STALL	EX	MEM	WB
8	ld																	IF	ID
9																			

Then that should be 16 cycle in one loop and 18 in last loop. More over, loop times will be 396/4=99. So the answer should be (98*16)+18=1586 cycles in the whole loop.

c. Chart has shown the changing. Then that should be 9 cycle in one loop and 12 in last loop. More over, loop times will be 396/4=99. So the answer should be (98*9)+12=894 cycles in the whole loop.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	ld	IF	ID	EX	MEM	WB									
3	addi		IF	ID	STALL	EX	MEM	WB							
4	sd			IF	STALL	ID	EX	MEM	WB						
5	addi					IF	ID	EX	MEM	WB					
6	sub						IF	ID	EX	MEM	WB				
7	bnez							IF	STALL	ID	EX	MEM	WB		
8	ld									IF	STALL	STALL	STALL	STALL	
9											IF	ID	EX	MEM	WB
10															
11															

d. Chart has shown the changing. Then that should be 8 cycle in one loop and 12 in last loop. More over, loop times will be 396/4=99. So the answer should be (98*8)+12=796 cycles in the whole loop.

D16

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	ld	IF	ID	EX	MEM	WB									
3	addi		IF	ID	STALL	EX	MEM	WB							
4	sd			IF	STALL	ID	EX	MEM	WB						
5	addi					IF	ID	EX	MEM	WB					
6	sub						IF	ID	EX	MEM	WB				
7	bnez							IF	STALL	ID	EX	MEM	WB		
8	ld									IF	ID	EX	MEM	WB	
9															
10															
11															

e. Chart has shown the changing. Then that should be 10 cycle in one loop and 19 in last loop. More over, loop times will be 396/4=99. So the answer should be (98*10)+19=999 cycles in the whole loop.

C.2

a.

control flow type	frequency/ins(percentage)	stalls(cycles)
sjump/call	1%	1
taken	60%*15%=9%	2
not taken	40%*15%=6%	1

speedup=pipeline depth/(1+pipeline stall cycles per instruction)

so speedup of ideal=4/(1+（1+0）)=4

speedup of real stalls=1x1%+2x9%+1x6%=0.25%

speedup of real=0.4/(1+0.25)=3.2

speedup of pipeline=4/3.2-1=25% will be reached without hazards

b. Similar to section a,

we have speedup of real stalls=4x1%+9x9%+8x6%=1.33

speedup of real=4/(1+1.33)=1.72

speedup of pipeline=4/1.72-1=33%

C.7

a. Referring equation

CPI=avg(stall cycles)/instruction+CPI of ideal

speedup=CPI of unpipeline x cycle time of unpipeline/(CPI of pipeline x cycle time of pipeline)

Because we can calculate the CPI of 5-stage is 6/5, the CPI of 12-stage is 11/8. Moreover, we have cycle time of 5-stage is 1 ns and 12 stage's time is 0.6 ns.

So we have speed up=(6/5x1)/(11/8x0.6)-1=45%

b. Referring equation

CPI=CPI of 5/12-stage +5/12-stage cycle time x number of cycles x instruction of 5/12-stage

CPI of 5=6/5+0.05 x 2 x 20/100= 1.22

CPI of 12 =11/8+0.05 x 5 x 20/100=14.425

so we have speed up=(1.22 x 1 x 1)/(1.425 x 0.6 x 1) - 1 =43%

3.1

instruction	number of cycles
LOOP	1+3=4
I0	1+4=5
I1	1+10=11
I2	1+3=4
I3	1+3=4
I4	1+2=3
I5	1+1=2
I6	1
I7	1
I8	1
I9	1+1=2

The baseline is shown in chart. Every instruction needs one clock cycle for EX and its latency of single cycle. So there will be 37 cycles per iteration.

3.2

instruction	number of cycles
LOOP	1+3=4
I0 WAIT FLD	1+4=5
I1 WAIT FMULT	1+(10)=1
I2	1+(3)+3(stall of LD)=4
I3 WAIT FLD	1+(2)+5(stall of DIVD)=6
I4 WAIT DIVD	1+(2)=1
I5	1+(1)=1
I6	1
I7	1
I8	1
I9	1+1=2

Because of detecting, it can let instruction do its action when in cycles of extra latency. So the answer should be sum of the cycle=27 cycles

3.3

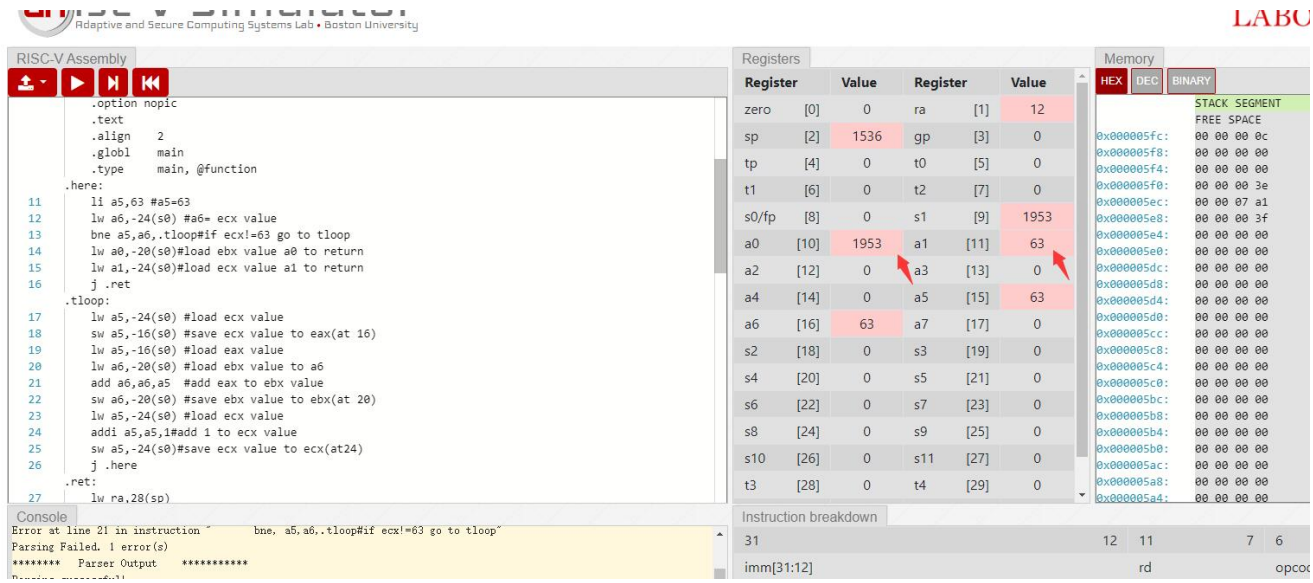
NO OF CYCLE / NO OF PIP	PIPELINE1	PIPELINE2
1	LOOP:FLD	NOP
2	STALL OF LD	NOP
3	STALL OF LD	NOP
4	STALL OF LD	NOP
5	I0:FMULTD	NOP
6	STALL OF MUL	NOP
7	STALL OF MUL	NOP
8	STALL OF MUL	NOP
9	STALL OF MUL	NOP
10	I1:FDIVD	I2:FLD
11	STALL OF LD	NOP
12	STALL OF LD	NOP
13	STALL OF LD	NOP
14	I3:ADDD	NOP
15	STALL OF DIVD	NOP
16	STALL OF DIVD	NOP
17	STALL OF DIVD	NOP
18	STALL OF DIVD	NOP
19	STALL OF DIVD	NOP
20	STALL OF DIVD	NOP
21	I4:ADDD	I5:FSD
22	I6:ADDI	I7:ADDI
23	I8:SUB	I9:BNZ
24	STALL OF BNZ	NOP

Because some instruction can execute when there are no hazard, we have the result of 24 cycles every iteration.

Part D:

C code:

```
int main(){
    int ebx=0;
    int ecx=0;
    int b;
    while(ecx!=63){
        eax=ecx;
        ebx=ebx+eax;
        ecx++;
    }
    return 0;
}
```



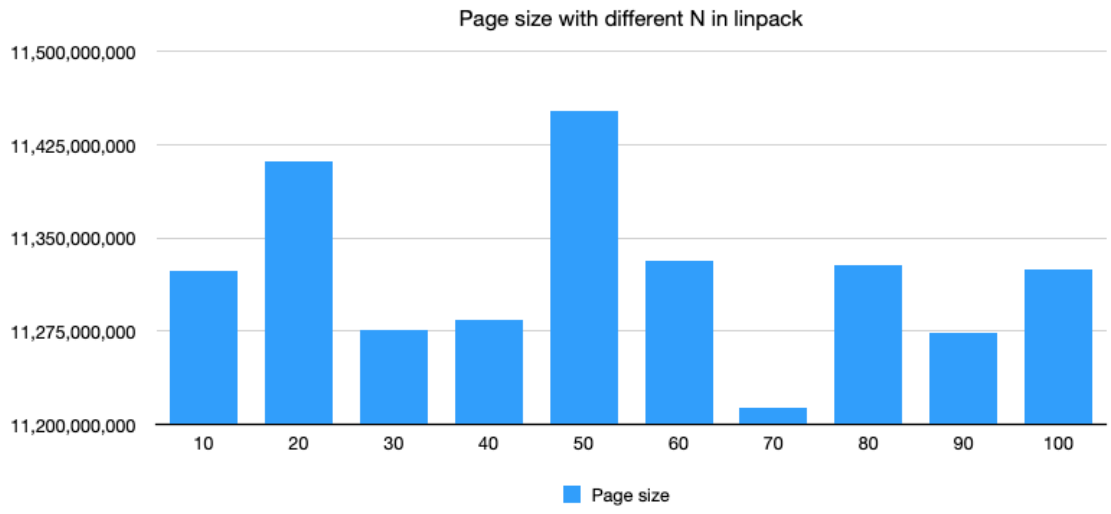
As we can see, this is the sum of 1 to 62, which is exactly 1953 as EBX, and 63 iteration stopping value as ECX as we got in the simulator.

Part E:

I am using pin tool to trace. For memory reading and writing, I used pintrace.cpp in manual example. It can trace memory addresses that have been read or writen. (The result and program can be seen in file pintrace)

I defined N as 10 to 100 and 10 as the gap. And I wrote a cpp program to read the max and min address. And then calculate the difference between them. Finally I divide them with 4096 to get the page size.

```
^C
zhangmingzhe@mingzhe-macmini pintrace % g++ readfile.cpp -o readfile.out
zhangmingzhe@mingzhe-macmini pintrace % ./readfile.out
max address= 7ffef5b7c7c8
min address= 55d03fb76040
The page size of N = 10 is 2a2eb6006
max address= 7ffca75ee7c8
min address= 5579d57c7040
The page size of N = 20 is 2a82d1e27
max address= 7fff1bffa7c8
min address= 55fd87d35040
The page size of N = 30 is 2a01942c5
max address= 7ffeed7dd7c8
min address= 55f5eb6c6040
The page size of N = 40 is 2a0902117
max address= 7fff109eb7c8
min address= 555583d06040
The page size of N = 50 is 2aa98cce5
max address= 7ffd2210c7c8
min address= 55c6562c8040
The page size of N = 60 is 2a36cbe44
max address= 7ffea90b67c8
min address= 563955c0f040
The page size of N = 70 is 29c5534a7
max address= 7ffd38c457c8
min address= 55c96d130040
The page size of N = 80 is 2a33cbb15
max address= 7ffe562677c8
min address= 55fe6fe46040
The page size of N = 90 is 29ffe6421
max address= 7ffff3fab7c8
min address= 55cfa1070040
The page size of N = 100 is 2a3052f3b
zhangmingzhe@mingzhe-macmini pintrace %
```



It seems page size is not relevant with N value in linpack.