Computer Architectures
Exam of September 11, 2020
Part 1

Exe #1, Point #1

Let consider the Branch Prediction Mechanism based on the Branch History Table (BHT).

You are requested to

1. *Describe the architecture of a BHT*

The BHT is a table composed of N elements of M bits each (typically, 1 or 2).

Exe #1, Point #2

2. *Describe in details the behavior of a BHT: when it is accessed, which input and output information are involved with each access*

Each time a processor decodes a conditional branch instruction, it accesses the BHT using the lowest bits of its address (ignoring those related to the instruction alignment). The selected BHT entry tells whether the instruction should be predicted Taken or NotTaken, based on previous executions of the same instruction. When the results of the branch is known, the BHT is possibly updated.

Exe #1, Point #3

3. *Assuming that the processor uses 32 bit addresses, each instruction is 4 byte wide, and the BHT is composed of 16 entries, clarify the content and size of the fields composing each BHT entry.*

The BHT in this case is composed of 16 entries, each storing 1 bit.

Exe #1, Point #4:
BHT initial content

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

Exe #1, Point #4: BTB content after l.d execution

l.d address	
0x00A50050	0000 0000 1010 0101 0000 0000 01**01 00**00

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

Exe #1, Point #4: BTB content after bnez execution

bnez address	
0x00A50054	0000 0000 1010 0101 0000 0000 01**01 01**00

0
0
0
0
0
1
0
0
0
0
0
0
0
0
0
0

Entry #5 is accessed. The entry value was 0 (Predict NotTaken) → Misprediction

Exe #1, Point #4: BTB content after addi execution

addi address	
0x00A60050	0000 0000 1010 0101 0000 0000 0101 0000

0
0
0
0
0
1
0
0
0
0
0
0
0
0
0
0

Exe #1, Point #4: BTB content after bez execution

bez address	
0x00A60054	0000 0000 1010 0101 0000 0000 01**01 01**00

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

Entry #5 is accessed.
The entry value was 1
(Predict Taken) →
Misprediction

Exe #1, Point #4: BTB content after andi execution

andi address	
0x00A60058	0000 0000 1010 0101 0000 0000 0101 1000

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

Exe #1, Point #4: BTB content after bez execution

bez address	
0x00A6005C	0000 0000 1010 0101 0000 0000 01**01 11**00

0
0
0
0
0
0
0
1
0
0
0
0
0
0
0
0

Entry #7 is accessed.
The entry value was 0
(Predict NotTaken) → Misprediction

Exe #1, Point #4: BTB content after add execution

add address	
0x00BB0040	0000 0000 1011 1011 0000 0000 **0100 0000**

0
0
0
0
0
0
0
1
0
0
0
0
0
0
0
0

Exe #1, Point #4: BTB content after bez execution

bez address	
0x00BB0044	0000 0000 1011 1011 0000 0000 01**00 01**00

0
1
0
0
0
0
0
1
0
0
0
0
0
0
0
0

Entry #1 is accessed. The entry value was 0 (Predict NotTaken) → Misprediction

Exe #2

Let consider a MIPS64 architecture including the following functional units (for each unit the number of clock periods to complete one instruction is reported):

- Integer ALU: 1 clock period
- Data memory: 1 clock period
- FP arithmetic unit: 2 clock periods (pipelined)
- FP multiplier unit: 4 clock periods (pipelined)
- FP divider unit: 8 clock periods (unpipelined)

You should also assume that

- The branch delay slot corresponds to 1 clock cycle, and the branch delay slot is not enabled
- Data forwarding is enabled
- The EXE phase can be completed out-of-order.

You should consider the following code fragment and, filling the following tables, determine the pipeline behavior in each clock period, as well as the total number of clock periods required to execute the fragment. The values of the constants k1 and k2 are written in f10 and f11 before the beginning of the code fragment.

```
; ******************** MIPS64 **********************
;  for (i = 0; i < 10; i++) {
;                     v5[i] = v1[i]*k1 + v2[i]/k2;
;  }
```

Exe #2

.data	Comments	Clock cycles
v1: .double “10 values”		
v2: .double “10 values”		
V3: .double “10 values”		
.text		
main: daddui r1,r0,0	r1← pointer	5
daddui r2,r0,10	r2 <= 20	1
loop: l.d f1,v1(r1)	f1 <= v1[i]	1
mul.d f2, f1, f10	f2 <= v1[i] *k1	5
l.d f3,v2(r1)	f3 <= v2[i]	0
div.d f4, f3, f11	f4 <= v2[i]/k2	7
add.d f5, f4, f2	f5 <= v1[i]*k1 + v2[i] /k2	2
s.d f5,v3(r1)	v3[i] <= f5	1
daddui r1,r1,8	r1 <= r1 + 8	1
daddi r2,r2,-1	r2 <= r2 - 1	1
bnez r2,loop		2
halt		1
total		216

Exe #2

main: daddui r1,r0,0	F	D	E	M	W																							5
daddui r2,r0,10		F	D	E	M	W																						1
loop: l.d f1,v1(r1)			F	D	E	M	W																					1
mul.d f2, f1, f10				F	D		E	E	E	E	M	W																5
l.d f3,v2(r1)					F		D	E	M	W																		0
div.d f4, f3, f11							F	D		E	E	E	E	E	E	E	E	M	W									7
add.d f5, f4, f2								F		D								E	E	M	W							2
s.d f5,v3(r1)										F								D	E		M	W						1
daddui r1,r1,8																		F	D		E	M	W					1
daddi r2,r2,-1																			F		D	E	M	W				1
bnez r2,loop																					F		D	E	M	W		2
halt																												1