CSPC 31 :
Computer Architecture

Assignment

Roll no. : **106119100**

Name : **Rajneesh Pandey**

Section : **CSE-B**

26/11/2021

CSPC 31 - C.A.

106119100
Rajneesh Pandey

Assignment

Question (1)

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}
I_1	IF	OF	PO	PO	PO	WB									
I_2		IF	OF	–	–	PO	PO	PO	PO	(PO)	WB				
I_3			IF	–	–	–	–	–	–	–	(OF)	[PO]	WB		
I_4				–	–	–	–	–	–	–	IF	–	[OF]	PO	WB

As,

It is mentioned in the question that operand forwarding takes place from PO stage to OF stage.

But since operand forwarding is from PO-OF, we can do like make the PO stage produce the output during the rising edge the clock and OF stage fetch the output during the falling edge. This would mean the final PO stage and OF stage can be done in one clock cycle making the total number of cycle = 13. And 13 is correct.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}
I_1	IF	OF	PO	PO	PO	WB							
I_2		IF	OF			PO	PO	PO	PO	PO	WB		
I_3			IF							OF	PO	WB	
I_4										IF	OF	PO	WB

Question ②

The miss rate (either local or global) for the first-level cache is 40/1000 or 4%.

The local miss rate for the second-level cache is 20/40 or 50%

The global miss rate of the second-level cache is 20/1000 or 2%.

Then,

$$\text{Average memory Access time} = \text{Hit time}_{L_1} + \text{Miss rate}_{L_1} \times (\text{Hit time}_{L_2} + \text{Miss rate}_{L_2} \times \text{Miss penalty}_{L_2})$$

$$= 1 + 4\% \times (10 + 50\% \times 200) = 1 + 4\% \times 110$$

$$= 5.4 \text{ clock cycle}$$

To see how many missers we get per instruction, we divide 1000 memory reference by 105 memory reference per instruction, which yields 607 instruction

Thus, we need to multiply the misses by 1.5 to get the number of misses per 1000 instruction.

we have 40 x 1.5 or 60 L1 misses, and 20 x 1.5 or 30 L2 misses, per 1000 instruction.

for average memory stall per instruction, assuming the misses are distributed uniformly between instruction and data.

Average memory small per instruction = Misses per instruction$_{L1}$ x Hit Time$_{L2}$ + Misses per inst$_{L2}$ x Miss penality$_{L2}$

If we subtract the L1 hit time from the average memory access time (AMAT) and then multiply by the average number of memory references per instruction, we get the same average memory stalls per instruction.

106119100

Question (3)

(a)

Amdahl's Law states:

Execution time after improvement =

(Execution time affected by improvement) / (Amount of Improvement) + Execution time unaffected

Assuming, initially that the floating point multiply, floating point divide and the other instruction had the same CPI,

Execution time after improvement with Divide = (20)/3 + (50+30) = 86.67

Execution time after improvement with multiply = 50/8 + (20+30) = 66.67

The management's goal can be met by making the improvement with Multiply alone

(b) If we make both the improvements,

Execution time after improvement

$= (50)/8 + \frac{20}{3} + 30$

$= 53.33$

The speedup relative to the original Machine = 100/53.33 = 1.88.

106119100

Question (4)

single processor.

Phase 1: serial,
runs for 4 minutes $\Rightarrow t_{phase1} = 4\,min$

Phase 2: parallelizable
runs for 8 min.
$\Rightarrow t_{phase2} = 8\,min$

No. of CPU given = 4. for parallel computer,

So,

$$speedup = \frac{1}{\left(\frac{t_{phase1}}{12} + \frac{(t_{phase2})}{12}/4\right)} = \frac{12}{6} = 2.$$

$$= \frac{1}{(4/12 + 8/12/4)} = 12/6 = 2$$

Question (5)

(a)

Avg. CPI for A = $4 \times 0.15 + 1 \times 0.25 + 5 \times 0.20 + 2 \times 0.40 = \underline{2.65}$

Avg CPI for B = $1 \times 0.20 + 3 \times 0.30 + 1 \times 0.15 + 5 \times 0.35$

$= \underline{3.0}$

(b) On which machine is the program faster with respect to

i. Execution time.

A = 2.65 * 60 = 159 billion clock cycles.

B = 3 * 80 = 240 billion clock cycle.

Exec time of A = $\frac{159}{2}$ = 79.5 s

Exec time of B = 240 / 2.5 = 96s

ii. MIPS rating

A = (60 * 10^9) / (79.5 * 10^6) = 755

B = (80 * 10^9) / (96 * 10^6) = 833.

Question (6)

The maximum number of memory references each cycle is 192 : 32 processors time 6 reference per precessor. Each SRAM bank is busy for 15/2.167 = 6.92 clock cycle, which we round up to 7 processor clock cycle.

Therefore, we require a minimum of 192x7 = 1344 memory banks!

The Cray T932 actually has 1024 memory banks, so the early models could not sustain full banduith to all processors simultaneously memory banks, so the

A subsequent memory upgrade replaced the 15ns asynchronous SRAMs with pipelined Syn-chronous SRAMs that more than halved the memory cycle time, thereby providing sufficient bandwidth.

Question (7)

There are two different dependences.

(1) S_1 uses a value computed by S_1 in an earlier iteration, since iteration "i" computes A[i+1], which is read in iteration i+1. The same is true of S_2 for B[i] and B[i+1].

(2) S_2 uses the value A[i+1] computed by S_1 in the same iteration.

These two dependences are different and have different effects. To see how they differ, lets assume that only one of these dependencies exist at a time.

Because the dependence of statement S_1 is on an earlier iteration of S_1. this dependence is loop carried. This dependence forces successive

iteration of this loop to execute in series.
The second dependence (S2 depending on S1) is within an iteration and is not loop carried. Thus, if this were the only dependence, multiple iterations of the loop could execute in parallel, as long as each pair of statements in an iteration were kept in order.

These intra loop dependence are common; for example a sequence of vector instruction that uses chaining exhibits exactly this sort of dependence.

Question (8)

Mips code :

```
L.D      F0, a          ; load scalar a
DADDIU   R4, Rx, #512   ; last address to load
L.D      F2, 0(Rx)      ; load X(i)
MUL.D    F2, F2, F0     ; a x X(i)
L.D      F4, 0(Ry)      ; load Y(i)
ADD.D    F4, F4, F2     ; a x X(i) + Y(i)
S.D      0(Ry), F4      ; store into Y(i)
DADDIU   Rx, Rx, #8     ; increment index to X
DADDIU   Ry, Ry, #8     ; increment index to Y
DSUBU    R20, R4, Rx    ; compute bound
BNEZ     R20, Loop      ; check if done
```

VMIPS code ofor DAXPY :

```
L.D.     F0, a          ; load scalar a
L.V      V1, Rx         ; load vector X
MULVS.D  V2, V1, F0     ; vector-scalar multiply
LV       V3, Ry         ; load vector Y
ADDV.D   V4, V2, V3     ; add
SV       Ry, V4         ; store the result
```

Question (9)

The peak single-precision floating-point throughput is

$$= 1.5 \times 16 \times 32 = 768 \text{ GFLOPS/s}$$

However, assuming each single precision operation require four byte two operands and output one four-byte result, sustaining this throughput (assuming no temporal locality). would require memory bandwidth of

$$12 \text{ bytes/FLOP} \times 768 \text{ GFLOPs/S} = 9.216 \text{ TB/s}.$$

Since 9.216 TB/s >> 100 GiB/s, this throughput is not sustainable, but can still be achieved in short bursts when using on-chip cache.

Question (10) :

The first convoy is occupied by the first LV instruction. The MULVS.D is dependendent on the first L.V. so it cannot be in the same convoy. The second LV instruction. can be in the same convoy as the MULVS.D.

The ADDV.D is dependent on the seond LV, so it must go in a following convoy.

1. LV
2. MULVS.D LV
3. ADDV.D
4. SV

The sequence requires four convoys and hence takes four chimes. Since the sequence takes a total of four chimes and there are two floating-point operation per result, the number of cycles per FLOP is 2 (ignoring any vector instruct) issue overhead).

Note, that although we allow the MULVS.D and the LV both to execute in convoy 2, most vector machine will take 2 clock cycle to initiate the instruction.