Advanced Computer Architectures
Exercises

Christian Rossi

Academic Year 2023-2024

Abstract

The course topics are:

- Review of basic computer architecture: the RISC approach and pipelining, the memory hierarchy.
- Basic performance evaluation metrics of computer architectures.
- Techniques for performance optimization: processor and memory.
- Instruction level parallelism: static and dynamic scheduling; superscalar architectures: principles and problems; VLIW (Very Long Instruction Word) architectures, examples of architecture families.
- Thread-level parallelism.
- Multiprocessors and multicore systems: taxonomy, topologies, communication management, memory management, cache coherency protocols, example of architectures.
- Stream processors and vector processors; Graphic Processors, GP-GPUs, heterogeneous architectures.

Contents

CHAPTER 1

Exercise session I

1.1 Exercise one

Assessing the impact of modifications on performance:

1. Substituting a hardware component with a faster alternative.
2. Incorporating multiple parallel systems for executing independent tasks.

Solution

1. By scaling up: response time decreases, while throughput increases.
2. Through scaling out: throughput experiences an increase. Response time will only escalate if a queue was present, awaiting computing resources.

1.2 Exercise two

Let's consider two CPUs: CPU1 and CPU2. CPU1 operates with a clock cycle of $2\,ns$, while CPU2 has an operating frequency of $700\,MHz$. Given the frequencies of occurrence of instructions for both CPUs:

Operation type	Frequency	CPU1 cycle	CPU2 cycle
A	0.3	2	2
B	0.1	3	3
C	0.2	4	3
D	0.3	2	2
E	0.1	4	3

1. Calculate the average CPI for CPU1 and CPU2.
2. Determine which CPU is the fastest.

Solution

1. The CPI (Cycle Per Instruction) is calculated as:

$$\text{CPI} = \frac{\text{clock cycles}}{\text{instruction}}$$

The average CPI is then obtained by:

$$\sum_{i=1}^{n} \text{CPI}_i \cdot \text{F}_i$$

Where, $\text{F}_i = \frac{\text{I}_i}{\text{instruction count}}$. For CPU1:

$$\text{CPI}_1 = 0.3 \cdot 2 + 0.1 \cdot 3 + 0.2 \cdot 4 + 0.3 \cdot 2 + 0.1 \cdot 4 = 2.7$$

For CPU2:

$$\text{CPI}_2 = 0.3 \cdot 2 + 0.1 \cdot 3 + 0.2 \cdot 3 + 0.3 \cdot 2 + 0.1 \cdot 3 = 2.4$$

2. We have that:

$$\begin{aligned}
\frac{\text{EXE}_{\text{CPU1}}}{\text{EXE}_{\text{CPU2}}} &= \left(\frac{\text{IC}_1 \cdot \text{CPI}_1}{F_1}\right)\left(\frac{F_2}{\text{IC}_2 \cdot \text{CPI}_2}\right) \\
&= \frac{\text{IC}_1 \cdot \text{CPI}_1 \cdot F_2}{F_1 \cdot \text{IC}_2 \cdot \text{CPI}_2} \\
&= \frac{\text{CPI}_1 \cdot F_2}{F_1 \cdot \text{CPI}_2} \\
&= \frac{2.7 \cdot 700MHz}{2.4 \cdot 500MHz} \\
&= 1.575
\end{aligned}$$

Hence, CPU2 is approximately 1.575 times faster than CPU1.

1.3 Exercise three

The speed of image processing on FPGA is 2.86 times faster than on a CPU. The power consumption of an FPGA is $100\,W$, while that of the CPU is $30.85\,W$. We want to achieve a speedup of 2 with the addition of an FPGA.

Solution

To achieve a speedup of 2 with the addition of an FPGA, we use Amdahl's law:

$$S_{overall} = \frac{1}{(1 - F_{enhanced}) + \frac{F_{enhanced}}{S_{enhanced}}}$$

Given $S_{overall} = 2$, we solve for F_{enhanced}:

$$2 = \frac{1}{(1 - F_{enhanced}) + \frac{F_{enhanced}}{2.86}} \rightarrow F_{enhanced} = 0.768$$

Therefore, with 76.8% of the processing offloaded to the FPGA, a speedup of 2 can be achieved.

1.4 Exercise four

Consider the following assembly program:

```
LW      $1, OFF($2)
ADDI    $3, $1, 4
SUB     $4, $1, $2
ADDI    $2, $1, -8
SW      $4, OFF($2)
```

No optimizations are applied in the MIPS pipeline. The processor operates with a clock cycle of 2 ns.

- Draw the pipeline schema and highlight potential hazards.
- Illustrate the actual execution with stall cycles inserted.
- Calculate Instruction Count (IC), CPI, and MIPS.

Solution

1. The pipeline schema is:

Instruction	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
LW $1, OFF ($2)	F	D	E	M	W				
FDDI $3, $1, 4		F	D	E	M	W			
SUB $4, $1, $3			F	D	E	M	W		
ADDI $2, $1, -8				F	D	E	M	W	
SW $5, OFF ($2)					F	D	E	M	W

   The potential hazards are:

   - Instruction 1 writes to register $1, and instructions 2, 3, and 4 read from it.
   - Instruction 2 writes to register $3, and instruction 3 reads from it.
   - Instruction 4 writes to register $2, and instruction 5 reads from it.

2. The real execution with stall cycles inserted is:

Instruction	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
LW $1, OFF ($2)	F	D	E	M	W										
ADDI $3, $1, 4		F	S	S	D	E	M	W							
SUB $4, $1, $3					F	S	S	D	E	M	W				
ADDI $2, $1, -8								F	D	E	M	W			
SW $5, OFF ($2)									F	S	S	D	E	M	W

3. The performance metrics are:

$$\text{IC} = 5$$

$$\text{CPI} = \frac{\text{CCs}}{\text{IC}} = \frac{15}{5} = 3$$

$$\text{MIPS} = \frac{\text{clock frequency}}{\text{CPI} \cdot 10^6} = \frac{0.5 \cdot 10^9}{3 \cdot 10^6} = 166$$

CHAPTER 2

Exercise session II

2.1 Exercise one

Consider the given program:

```
i1: add $t1, $t0, $t1
i2: add $t2, $t1, $t2
i3: subi $t0, $t2, 1
i4: sw $t0, 0x00BB($t2)
i5: beq $t0, $t2, 0x0089
```

Assuming no forwarding, register file access with read/write optimization, and control hazards solved in the instruction decode stage (ID):

1. Define all conflicts/dependencies and analyze whether they cause hazards and the theoretical amount of stalls.
2. Draw the effective pipeline schema:
3. Draw the effective pipeline schema assuming EX/EX, MEM/EX, and MEM/MEM forwarding paths are available.
4. Draw the effective pipeline schema assuming that the previous forwarding paths with also EX/ID are available.

Solution

1. The potential issues are:

Instruction number	Instruction dependency	Register involved	Hazard	Stalls
i2	i1	$t1	yes	2
i3	i2	$t2	yes	2
i4	i2	$t2	yes	1
i4	i3	$t0	yes	2
i5	i2	$t2	no	0
i5	i3	$t0	yes	1

2. The requested pipeline schema with stalls is as follows:

Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15
add $t1, $t0, $t1	IF	ID	EX	M	WB										
add $t2, $t1, $t2		IF	S	S	ID	EX	M	WB							
subi $t0, $t2, 1			S	S	IF	S	S	ID	EX	M	WB				
sw $t0, 0×00BB ($t2)						S	S	IF	S	S	ID	EX	M	WB	
beq $t0, $t2, 0×0089									S	S	IF	ID	EX	M	WB

3. The requested pipeline schema with forwarding is as follows:

Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
add $t1, $t0, $t1	IF	ID	EX	M	WB					
add $t2, $t1, $t2		IF	ID	EX	M	WB				
subi $t0, $t2, 1			IF	ID	EX	M	WB			
sw $t0, 0×00BB ($t2)				IF	ID	EX	M	WB		
beq $t0, $t2, 0×0089					IF	S	ID	EX	M	WB

4. The requested pipeline schema with forwarding and EX/ID is:

Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9
add $t1, $t0, $t1	IF	ID	EX	M	WB				
add $t2, $t1, $t2		IF	ID	EX	M	WB			
subi $t0, $t2, 1			IF	ID	EX	M	WB		
sw $t0, 0×00BB ($t2)				IF	ID	EX	M	WB	
beq $t0, $t2, 0×0089					IF	ID	EX	M	WB

2.2 Exercise two

Explain (with effective support) the design of a 1-bit branch history table (1-BHT) and a 2-bit branch history Table (2-BHT) capable of executing the provided assembly code (with R0 set to 2000 and R1 set to 0).

```
LOOP:   LD F1 0 R0
        ADDD F2 F1 F1
        ADDI R1 R1 100
LOOP2:  MULTD F2 F2 F1
        SUBI R1 R1 1
        BNEZ R1 LOOP2
        SUBI R0 R0 2
        BNEZ R0 LOOP
```

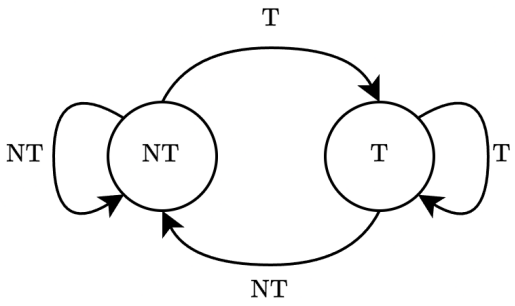
Compute the number of mispredictions in the various cases.

Solution

The outer loop iterates 1000 times because R0 starts at 2000 and decreases by two each iteration.

The inner loop iterates 100 times for each outer loop iteration because R1 starts at 100 and decreases by one each iteration. Consequently, the total iterations of the inner loop amount to $1000 \times 100 = 100000$.

For the one-bit branch table computation, a finite state machine can be employed, depicted below:



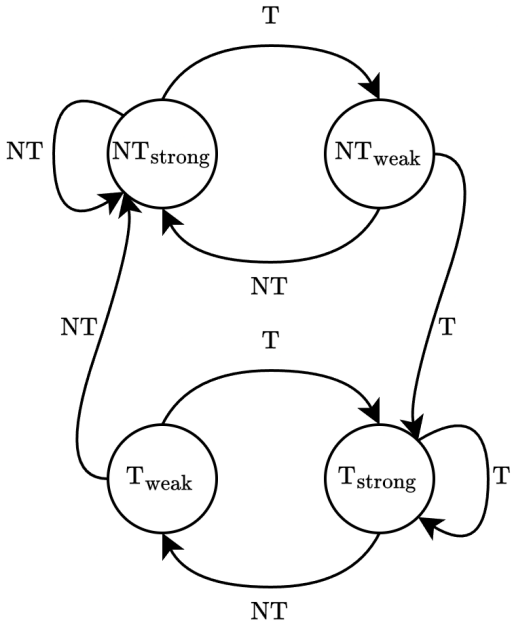
In the absence of collision, one bit can be assigned for each loop. This allows four possible initializations: T-T, T-NT, NT-T, NT-NT.

- For the first case (T-T), there are $1 + 1 + (1000 - 1) \times 2$ iterations.
- For the second case (T-NT), there are $1 + 1000 \times 2$ iterations.
- For the third case (NT-T), there are $2 + 1 + (1000 - 1) \times 2$ iterations.
- For the fourth case (NT-NT), there are $2 + 1000 \times 2$ iterations.

In the presence of collision, one bit is allocated for each loop. This yields two possible initializations: T, NT. The iterations for each case are calculated as follows:

- For the first case (T), there are $(1 + 1) \times (1000 - 1) + 1$ iterations.
- For the second case (NT), there are $1 + (1 + 1) \times (1000 - 1) + 1$ iterations.

The two-bit branch table can be designed using a finite state machine, as illustrated below:



In the absence of collision, where one bit is assigned for each loop, there are eight possible initializations. The worst cases are:

- $\mathrm{NT}_{\mathrm{weak}}, \mathrm{NT}_{\mathrm{weak}}$: resulting in 1000×2 mispredictions for LOOP2 and 2 mispredictions for LOOP.
- $\mathrm{NT}_{\mathrm{strong}}, \mathrm{NT}_{\mathrm{strong}}$: resulting in $3 + (1000 - 1) \times 1$ mispredictions for LOOP2 and 2 mispredictions for LOOP.

The best cases are:

- $\mathrm{T}_{\mathrm{weak}}, \mathrm{T}_{\mathrm{weak}}$: Resulting in $1 + (1000 - 1) \times 2$ mispredictions for LOOP2 and 1 for LOOP.
- $\mathrm{T}_{\mathrm{strong}}, \mathrm{T}_{\mathrm{strong}}$: Resulting in 1000×1 mispredictions for LOOP2 and 1 for LOOP.

Considering these cases, the iterations for each case are calculated accordingly:

- For the first case, we have $1 + 1 + (1000 - 1) \times 2$.
- For the second case, we have $1 + 1000 \times 2$.
- For the third case, we have $2 + 1 + (1000 - 1) \times 2$.
- For the fourth case, we have $2 + 1000 \times 2$.

In the presence of collision, where one bit is used for each loop, there are four possible initializations ($\mathrm{T}_{\mathrm{strong}}$, $\mathrm{T}_{\mathrm{weak}}$, TNstrong, NT : weak). The iterations for each case are calculated as follows:

- For the first case, we have $1 \times 1000 + 1$.
- For the second case, we have $1 \times 1000 + 1$.
- For the third case, we have $2 + 1 \times 1000 + 1$.
- For the fourth case, we have $1 + 1 \times 1000 + 1$.

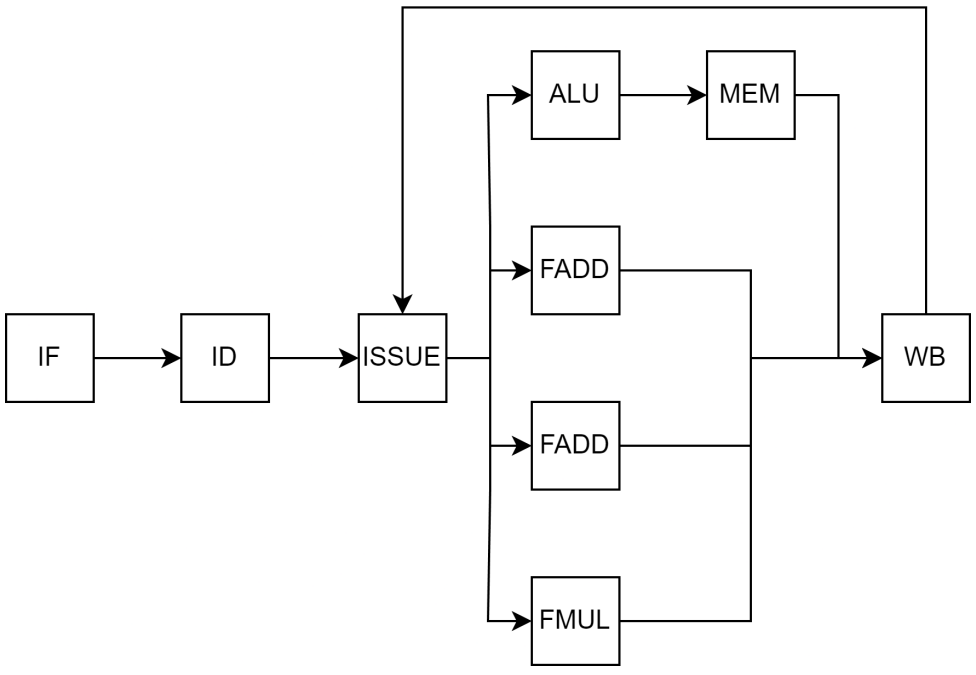
Comparing the worst-case scenario of the two-bit branch table with the best-case scenario of the one-bit branch table, it's evident that the worst 2BHT performs better than the best 1BHT.

CHAPTER 3

Exercise session III

3.1 Exercise one

In this problem, we'll analyze the execution of a code segment on a single-issue out-of-order processor.



Consider now the code:

```
I1 lw.d $F3,B($R0)
I2 add.d $F2,$F2,$F3
I3 mul.d $F5,$F4,$F4
I4 addi $R0,$R0,8
I5 lw.d $F3,B($R0)
I6 add.d $F2,$F3,$F5
```

Examine each conflict taking into account the following operation durations:

- Arithmetic logic unit operations: one cycle.
- Memory operations: three cycles.
- Floating point addition: three cycles.
- Floating point multiplication: five cycles.

Solution

The conflicts are:

- RAW dependencies between instructions one and two, involving register $F3.
- WAR conflict between instruction one and instructions one and four, concerning register $R0.
- WAW conflict between instruction one and instructions one and five, affecting register $F3.
- WAR conflict between instruction two and instructions two and five, regarding register $F3.
- RAW dependency between instructions four and five, involving register $R0.
- RAW dependency between instructions three and six, involving register $F5.
- WAW conflict between instruction two and instructions two and six, impacting register $F2.
- RAW dependency between instructions five and six, involving register $F3.

A possible execution is:

Instruction	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**	**C11**	**C12**	**C13**	**C14**	**C15**	**C16**	**C17**	**C18**	**C19**	**C20**
1	F	D	IS	E1	E2	E3	W													
2		F	D	S	S	S	IS	E1	E2	E3	W									
3			F	S	S	S	D	IS	E1	E2	E3	E4	E5	W						
4				S	S	S	F	D	IS	E	S	W								
5								F	S	D	S	IS	E1	E2	E3	W				
6									S	F	S	D	S	S	S	IS	E1	E2	E3	W

If we assume that the issue stage is a buffer with an unlimited capacity to hold instructions awaiting execution:

Instruction	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**	**C11**	**C12**	**C13**	**C14**	**C15**	**C16**	**C17**	**C18**	**C19**	**C20**
1	F	D	IS	E1	E2	E3	W													
2		F	D	S	S	S	IS	E1	E2	E3	W									
3			F	D	S	S	S	IS	E1	E2	E3	E4	E5	W						
4				D	S	D	S	S	IS	E	S	W								
5					S	F	S	S	S	D	S	IS	E1	E2	E3	W				
6							S	S	S	F	D	S	S	S	S	IS	E1	E2	E3	W