國立高雄大學資訊工程學系 計算機結構期中考考卷

姓名：　　　　　　　　　　　　　　　　學號：

1. (8%) Please explain the three kinds of hazards：
   A. Structural hazards
   B. Data hazards
   C. Control hazards

   Give two common methods for eliminating structure hazards.

   - Structural hazards: Hardware cannot support this combination of instructions - two instructions need the same resource.
   - Data hazards: Instruction depends on result of prior instruction still in the pipeline
   - Control hazards: Pipelining of branches & other instructions that change the PC

   - Duplicate resources　Pipeline the resource　Reorder the instructions

2. (9%) Please explain the three kinds of data hazards and give some examples：
   A. RAW
   B. WAR
   C. WAW

Read After Write (RAW)
InstrJ tries to read operand before InstrI writes it
I: add r1,r2,r3
J: sub r4,r1,r3

- Caused by a “Dependence” (in compiler nomenclature). This hazard results from an actual need for communication.

Write After Read (WAR)
InstrJ writes operand *before* InstrI reads it
I: sub r4,r1,r3

J: add r1,r2,r3
K: mul r6,r1,r7

- Called an "anti-dependence" by compiler writers.
  This results from reuse of the name "r1".
- WAR hazards can happen if instructions execute out of order or access data late

Write After Write (WAW)
InstrJ writes operand *before* InstrI writes it.
I: sub r1,r4,r3
J: add r1,r2,r3
K: mul r6,r1,r7

- Called an "output dependence" by compiler writers
  This also results from the reuse of name "r1".

3. (6%) Please compare the difference between dynamic scheduling and static scheduling.
   A. With static scheduling the compiler tries to reorder these instructions during compile time to reduce pipeline stalls.
      i. Uses less hardware
      ii. Can use more powerful algorithms
   B. With dynamic scheduling the hardware tries to rearrange the instructions during run-time to reduce pipeline stalls.
      i. Simpler compiler
      ii. Handles dependencies not known at compile time
      iii. Allows code compiled for a different machine to run efficiently.

4. (10%) Please compare Tomasulo Algorithm and Scoreboard. Please describe the drawbacks of Tomasulo Algorithm.
   A. Control & buffers distributed with Function Units (FU) vs. centralized in scoreboard;
      i. FU buffers called "reservation stations"; have pending operands
   B. Registers in instructions replaced by values or pointers to reservation stations(RS); called register renaming ;
      i. avoids WAR, WAW hazards
      ii. More reservation stations than registers, so can do optimizations compilers can't

C. Results to FU from RS, not through registers, over Common Data Bus that broadcasts results to all FUs

D. Load and Stores treated as FUs with RSs as well

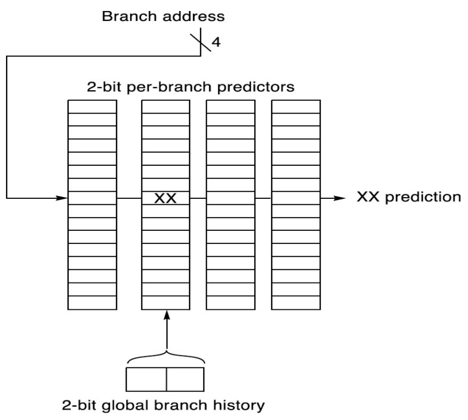
E. Integer instructions can go past branches, allowing FP ops beyond basic block in FP queue

Pipelined Functional Units	Multiple Functional Units
(6 load, 3 store, 3 +, 2 x/÷)	(1 load/store, 1 + , 2 x, 1 ÷)
window size: ≤ 14 instructions	≤ 5 instructions
No issue on structural hazard	same
WAR: renaming avoids	stall completion
WAW: renaming avoids	stall issue
Broadcast results from FU	Write/read registers
Control: reservation stations	central scoreboard

drawback

- The scheme requires a large amount of hardware
- Many associative stores (CDB) at high speed
- Performance limited by the single Common Data Bus
  - Multiple CDBs
    - Each CDB must interact with each reservation station
    - The associate tag-matching hardware would need to be duplicated at station for each CDB.

5. (12%) Please describe what is the Correlating Branch Prediction Buffer. Assume that there are 8K bits in the Correlating Branch Prediction Buffer. Please derive the numbers of entries in (0,1), (0,2), (2,2), and (12,2) predicators.



A. Use behavior of the last m branches
    i. Use last m branches = global branch history
    ii. Use n bit predictor
B. 2m n-bit predictors for each branch
C. Simple implementation
    i. Use m-bit shift register to record the behavior of the last m branches

- (0,1): 8K entries
- (0,2): 4K entries
- (2,2): 1K entries
- (12,2): 1 entry!

6. (17%) What is Instruction Set Architecture? Please explain the following ISAs and give examples.
    A. Accumulator
    B. Stack
    C. Memory-Memory
    D. Register-Memory
    E. Register-Register

Organization of Programmable Storage
-- Data Types & Data Structures:
    Encodings & Representations
-- Instruction Formats
-- Instruction (or Operation Code) Set
-- Modes of Addressing and Accessing Data Items and Instructions
-- Exceptional Conditions

Accumulator (before 1960):
    1 address add A   acc <- acc + mem[A]

Stack (1960s to 1970s):
    0 address add tos <- tos + next

Memory-Memory (1970s to 1980s):
    2 address add A, B mem[A] <- mem[A] + mem[B]
    3 address add A, B, C   mem[A] <- mem[B] + mem[C]

Register-Memory (1970s to present):

2 addressadd R1, A R1 <- R1 + mem[A]

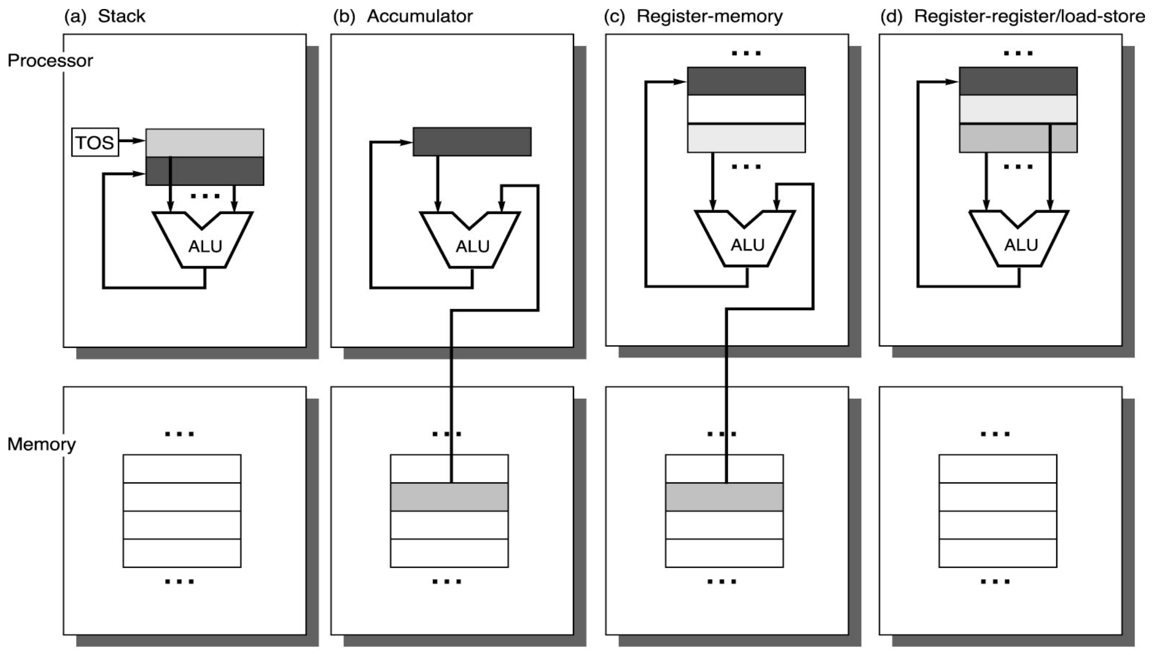
load R1, A R1 <_ mem[A]

Register-Register (Load/Store) (1960s to present):

3 address add R1, R2, R3 R1 <- R2 + R3

load R1, R2 R1 <- mem[R2]

store R1, R2 mem[R1] <- R2



7. (8%) What is Big Endian and Little Endian? What is Alignment?

A. The low-order byte of an object is stored in memory at the lowest address, and the high-order byte at the highest address. (The little end comes first.)

B. For example, a 4-byte object

i. (Byte3 Byte2 Byte1 Byte0)

ii. Base Address+0 Byte0

iii. Base Address+1 Byte1

iv. Base Address+2 Byte2

v. Base Address+3 Byte3

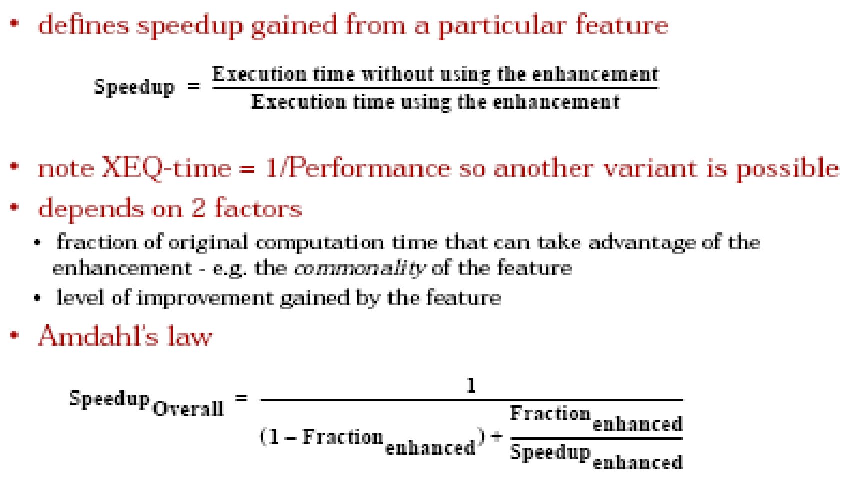
- The high-order byte of an object is stored in memory at the lowest address, and the low-order byte at the highest address. (The big end comes first.)

- For example, a 4-byte object
  - (Byte3 Byte2 Byte1 Byte0)
  - Base Address+0 Byte3
  - Base Address+1 Byte2
  - Base Address+2 Byte1
  - Base Address+3 Byte0

Alignment: objects must fall on address that is multiple of their size

8. (8%) Please describe Moore's Law. What is Amdahl's Law? Please explain it and give an example.

   A. ***# on transistors on cost-effective integrated circuit double every 18 months (IC 上可容納的電晶體數目，約每隔 18 個月便會增加一倍，性能也將提升一倍。)***



- defines speedup gained from a particular feature

$$\text{Speedup} = \frac{\text{Execution time without using the enhancement}}{\text{Execution time using the enhancement}}$$

- note XEQ-time = 1/Performance so another variant is possible
- depends on 2 factors
  - fraction of original computation time that can take advantage of the enhancement - e.g. the *commonality* of the feature
  - level of improvement gained by the feature
- Amdahl's law

$$\text{Speedup}_{\text{Overall}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

9. (8%) Explain how Tomasulo's algorithm handles WAW hazards and how this differs from the way scoreboarding handles WAW hazards.

- Presence of anti- and output- dependences
  - Lead to WAR and WAW stalls
    - Wait for WAR hazards
    - Prevent WAW hazards
- No forwarding hardware
- Since there can be more reservation stations than real registers, Tomasulo can eliminate hazard arising from name dependence.

Register renaming (to avoid WAR and WAW)

- Differences from Scoreboarding

- Distributed hazard detection and control (through reservation stations)
- Results are bypassed to function units
- Common data bus (CDB) broadcasts results to all FUs.
- HW renaming of registers to avoid WAR, WAW hazards
- Load and Stores treated as FUs as well
- Registers in instructions replaced by pointers to reservation station buffers

10. (8%) Explain how Tomasulo’s algorithm handles WAR hazards and how this differs from the way scoreboarding handles WAR hazards.

11. (20%) Please write the contents of Instruction Status, Reservation Stations, and Register result status at clock cycle 11.

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

				S1	S2	RS	RS
Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*
	Add1	No					
	Add2	No					
	Add3	No					
4	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
11	*FU*	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

Note：FP multiply (10 EX cycles)、FP add (2 EX cycles)、and FP divide (40 EX cycles)、load (2 EX cycles)