所別：資訊工程學系碩士班 不分組(一般生)　　科目：作業系統與計算機組織　　共 二 頁 第 一 頁
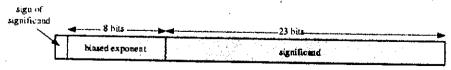
軟體工程研究所碩士班 不分組(一般生)

＊請在試卷答案卷（卡）內作答

＊本科考試禁用計算器

1. Fill in o(True) or ×(False)

1.1 General knowledge (6%)

(1). (__) Register is implemented inside the main memory on computer board

(2). (__) von Neumann architecture has data and instructions in the same memory space.

(3). (__) Static RAM is typically used to implement Cache

(4). (__) Dynamic RAM is faster than Static RAM in access time.

(5). (__) In modern computers, CPU and Memory are connected by BUS

(6). (__) Direct Mapping in Cache system is more complex in implementation than Set Associative Mapping

1.2 Amdahl's Law (3%)

(7). (__) A speedup of 40 on 40% of the program will result in an overall speedup of at least 2.

(8). (__) A speedup of 20 on 50% of the program will result in an overall speedup of at least 2.

(9). (__) A speedup of 10 on 60% of the program will result in an overall speedup of at least 2.

1.3 Processor Performance (3%)

(10). (__) Program execution time increases when the clock rate increases

(11). (__) Program execution time increases when the CPI increases

(12). (__) Program execution time increases when the instruction count (IC) increases

1.4 Pipelining (3%)

(13). (__) Splitting the shortest stage of a five-stage pipeline will result in a higher clock rate.

(14). (__) With single-issue, in-order execution, and the classical five-stage pipeline with no bypassing, WAW hazards never cause any "bubbles" (stalls) in the pipeline.

(15). (__) With a single-issue, in-order execution, and the classical five-stage pipeline with bypassing, RAW hazards never cause any "bubbles" (stalls) in the pipeline.

2. Let the decimal numbers A=54, B= -77, give their 8-bit 2's complement representation: (4%)

2.1 Compute A + B in 8-bit 2's complement

2.2 Compute A-B in 8-bit 2's complement, is it overflow ? why or why not ?

3. Processor control

3.1 Please describe clearly and compare microprogramming and hardwired design for controlling a processor, in which one instruction is executed in several clock cycles. (5%)

3.2 Given that a processor supports four types of instructions: Arithmetic, Memory Load, Memory Store and Branch and that the following truth table specifies the inputs and outputs of the control mechanism, what control design strategy will you use and why? Please show the details of your design. (6%)
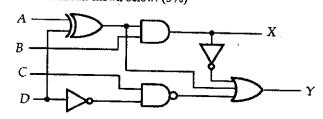
Control	Input						Output						
Signal Name	In5	In4	In3	In2	In1	In0	Reg	RegWrite	Mem Read	MemWrite	Branch	ALU1	ALUOp2
Arithmetic	0	0	0	0	0	0	1	1	0	0	0	1	0
Mem. Load	1	0	0	0	1	1	0	1	1	0	0	0	0
Mem. Store	1	0	1	0	1	1	x	0	0	1	0	0	0
Branch	0	0	0	1	0	0	x	0	0	0	1	0	1

4. For the 32 bit Floating point system given below, the exponent bias is 127, i.e., if the true exponent is 5, the biased exponent would be 132. Find the sign bit, exponent, and significand representations of floating point number 118.75. (3%)

5. Write a complete VHDL or Verilog code for the circuit shown below: (3%)

注意：背面有試題

6. Memory hierarchy

    6.1 What is TLB? What information will be stored in TLB? (4%)

    6.2 Given that a computer has a 32-bit virtual address with the page size equal to 4K bytes and a 28-bit physical address. What is the size of the page table in *bits*, if we ignore other flag bits? (4%)

    6.3 Following the previous question, after the address translation, a two-way set-associative cache with the block size equal to 64 bytes will be accessed. Assuming that the cache can hold 8K blocks, please draw the block diagram of accessing a block in this cache. You should show how the address will be divided into fields and demonstrate important components in this cache, including multiplexors, cache hit generation, data hits, valid/dirty bits and wires. (6%)

7. Suppose that two long running processes, *P1* and *P2*, are running in a system for different users, and there are no other processes in the system. Process *P1* consists of three threads and process *P2* consists of two threads. Assume that the threads never block and that the processes are entirely resident in primary memory. (8%)

    7.1 What percentage of CPU time will process *P1* get if the threads are kernel threads entirely?

    7.2 What percentage of CPU time will process *P2* get if the threads are user threads entirely?

8. Suppose that the following processes arrive for execution at the times indicated. Each process will run the listed amount of time.

    8.1 Assume that nonpreemptive scheduling is adopted. What is the average turnaround time for these processes with the FCFS scheduling algorithm? (3%)

    8.2 Assume that preemptive scheduling is adopted. What is the average waiting time for these processes with the SJF scheduling algorithm? (4%)

Process	Arrival Time	Burst Time
P1	0	8
P2	2	4
P3	4	2

9. Multiple choices question. Which statements in the following are correct? (10%)

    (a) Domain Name Service (DNS) can be used to acquire IP addresses.

    (b) Reverse Address Resolution Protocol (RARP) can be used to acquire IP addresses.

    (c) Network Address Translation (NAT) is used to map MAC addresses to IP addresses.

    (d) Multicasting is adopted in Dynamic Host Configuration Protocol (DHCP).

    (e) IPv6 addresses are 128 bits long.

    (f) All of the above.

    (g) None of the above.

10. Typically, at the completion of a device I/O, a single interrupt is raised and appropriately handled by the host processor. In certain settings, however the code that is to be executed at the completion of the I/O can be broken into two separate pieces, one of which executes immediately after the I/O completes and schedules a second interrupt for the remaining piece of code to be executed at a later time. (15%)

    10.1 What is the purpose of using this two pieces strategy in the design of interrupt handles?

    10.2 How to notice the scheduler to complete the remaining piece of code?

    10.3 In which cases mutual exclusive lock will be considered or not?

11. Using the program shown in the following list(10%),

    11.1 Explain what will be output at LINE A , LINE B ?

    11.2 Please give your reason.

```c
#include
<sys/types.h>
#include <stdio.h>
#include <unistd.h>
int value   = 10;
int main( )
{
    pid_t    pid;
    pid = fork ( );
    if (pid == 0 ) {
        value +=15;
        printf("CHILD: value = %d", value); /* LINE A */
    }
    else if (pid > 0) {
        wait (NULL);
        value +=3;
        printf("PARENT: value = %d", value); /* LINE B */
        exit(0);
    }
}
```