Problem 1: (20 points)

What is the "printf" output from the code below? Write the answer in the box below

```
#include<stdio.h>
#include<wait.h>
#include<signal.h>
pid_t pid;
int counter = 0;
void handler1(int sig)
{
    counter++;
    printf("counter = %d\n", counter);
    /* Flushes the printed string to stdout */
    fflush(stdout);
    kill(pid, SIGUSR1);
}
void handler2(int sig)
{
    counter += 3;
    printf("counter = %d\n", counter);
    exit(0);
}

int main()
{
    pid_t p;
    int status;
    signal(SIGUSR1, handler1);
    if ((pid = fork()) == 0)
    {
        signal(SIGUSR1, handler2);
        kill(getppid(), SIGUSR1);
        while(1) ;
    }
    if ((p = wait(&status)) > 0)
    {
        counter += 4;
        printf("counter = %d\n", counter);
    }
```

counter = 1 //(parent's handler)

counter = 3 //(child's handler)

counter = 5 //(parent's main)

Problem 2: (10 points)

What is the "printf" output from the code below? Write the answer in the box below

```
main() {

    fork();

    fork();

    fork();

    printf("hello world\n");
}
```

The main() will print one time and creates 3 children, let us say Child_1, Child_2, Child_3.

All of them printed once.

The Child_3 will not create any child.

Child2 will create one child and that child will print once.

Child_1 will create two children, say Child_4 and Child_5 and each of them will print once.

Child_4 will again create another child and that child will print one time.

A total of eight times the printf statement will be executed.

Here is the code running on the ZedBoard:

```
root@master_18_04:~/code/ZED/EXAM_1_Q2# ./Question_2
hello world
hello world
hello world
hello world
hello world
root@master_18_04:~/code/ZED/EXAM_1_Q2#hello world
hello world
hello world
```

Notice that the printf output overruns the command line prompt
To prevent this you need to use a fflush(stdout) command as shown above in Question 1

Problem 3: (14 points) -- Write your answer in the box to the right of the question.

1. The time taken to respond to an interrupt is known as
   a) Interrupt delay
   b) Interrupt time
   c) Interrupt latency <-------- This is the preferred answer will take a, c, e ☺
   d) A & B & C
   e) A & C only

E

2. Which part of an interrupt service routine performs tasks in direct response to interrupts?
   a) Bottom half <---------- Part of the ISR that is in user space
   b) Top half <---------- Part of the ISR that is in kernel space
   c) ISR <---------- Composed of top and bottom half code
   d) A & B
   e) A & B & C

B

3. Which of the following are asynchronous to the operation of a processing system?
   a) Exceptions
   b) Software
   c) Interrupts <---------- This is only thing that is asynchronous to the PS
   d) Memory
   e) GPIO
   f) A & C

C

4. Which software interrupt is used in ARM?
   a) TTC interrupt
   b) SVC <-------- This replaces the SWI instruction
   c) SWI <-------- Still valid in the ARM assembler….
   d) NMI <-------- This is a hardware interrupt
   e) B & C
   f) B & C & D

E

5. What does NMI stand for?
   a) non-machine interrupt
   b) non-maskable interrupt <-------- This interrupt cannot be blocked by the HW
   c) non-massive interrupt
   d) non-memory interrupt
   e) None of the above

B

6. How many potential interrupts does the ARM SWI support?
   a) 2^{16}
   b) 2^{32}
   c) 2^{24} <---------- There are 2^{24} encodings but only one interrupt
   d) Only one

D

7. At which point the processor will start to internally recognize and process an interrupt?
   a) interrupt pointer
   b) instruction pointer
   c) instruction boundary <-------- Instructions have to complete before recognizing INTs
   d) interrupt boundary

C

Problem 4: (26 points) -- Write your answer in the box to the right of the question.

DEFINITIONS:

PROCESS: IS THE SET OF CODE/INSTRUCTIONS WHICH OPERATES ON RELATED DATA. THE PROCESS HAS ITS OWN STATE, I.E., SLEEPING, RUNNING, STOPPED ETC. WHEN A PROGRAM GETS LOADED INTO MEMORY IT BECOMES PROCESS. EACH PROCESS HAS AT LEAST ONE THREAD WHEN CPU IS ALLOCATED CALLED SINGLE THREADED PROGRAM.

THREAD: IS A PORTION OF THE PROCESS. MORE THAN ONE THREAD CAN EXIST AS PART OF PROCESS. THREAD HAS ITS OWN PROGRAM AREA AND MEMORY AREA. MULTIPLE THREADS INSIDE ONE PROCESS CANNOT ACCESS EACH OTHER DATA. PROCESS HAS TO HANDLE SYNCHRONIZATION OF THREADS TO ACHIEVE THE DESIRABLE BEHAVIOR.

TASK: THE TERM *"TASK"* IS MOSTLY USED IN THE CONTEXT OF SCHEDULING , WHERE IT CAN REFER TO EITHER A *THREAD* OR A *PROCESS*, THAT CAN BE SCHEDULED TO RUN ON A PROCESSOR. TASK AND PROCESS ARE CONSIDERED TO BE SYNONYMS

1. WHICH OF THE FOLLOWING MAKES AN APPLICATION PROGRAM HARDWARE INDEPENDENT?
   - A) SOFTWARE
   - B) APPLICATION MANAGER
   - C) OPERATING SYSTEM <-------- THIS IS ONLY THING THAT ALLOWS PORTABILITY OF YOUR CODE
   - D) KERNEL
   - E) ISR

   C

2. WHICH FORMS THE HEART OF THE OPERATING SYSTEM?
   - A) STARTUP.S
   - B) APPLICATIONS
   - C) HARDWARE
   - D) OPERATING SYSTEM
   - E) KERNEL <-------- THIS IS THE HEART OF ANY SYSTEM.

   E

3. WHICH OF THE FOLLOWING WORKS BY DIVIDING THE PROCESSOR'S TIME?
   - A) CYCLIC EXECUTIVE <-------- THIS IS GENERALLY A ROUND ROBIN SCHEDULER
   - B) MULTI-TASKING OPERATING SYSTEM <--------- THIS DOES THE REAL WORK
   - C) KERNEL <-------- THIS DOES THE HW PIECE I.E., TIMERS, ETC.
   - D) APPLICATION THREADS

   B

4. WHICH OF THE FOLLOWING CONTROLS THE TIME SLICING MECHANISM IN A MULTITASKING OPERATING SYSTEM?
   - A) KERNEL
   - B) SINGLE TASKING KERNEL
   - C) MULTITASKING KERNEL
   - D) APPLICATION MANAGER
   - E) "WATCHDOG" TIMER

   C

5. WHICH OF THE FOLLOWING CAN PERIODICALLY TRIGGER THE CONTEXT SWITCH?
   - A) SOFTWARE INTERRUPT
   - B) HARDWARE INTERRUPT <---- THIS IS THE ONLY MECHANISM WE USE TO TRIGGER A CONTEXT SWITCH
   - C) GPIO INTERRUPT
   - D) MEMORY INTERRUPT
   - E) A & B
   - F) ALL OF THE ABOVE

   B

6. WHICH OF THE FOLLOWING STORES ALL THE TASK INFORMATION THAT THE SYSTEM REQUIRES?
   - A) TASK ACCESS BLOCK
   - B) GENERAL PURPOSE REGISTERS
   - C) ACCESS CONTROL LIST (ACL)
   - D) TASK CONTROL BLOCK

   D

7. Which determines the sequence and the associated task's priority?
   a) Scheduling algorithm
   b) Task ready list
   c) Task control block
   d) A & B
   e) A & C

A

8. Which controls the memory sharing between tasks?
   a) Kernel <-------- The kernel controls the HW in the system i.e., memory.
   b) Application code
   c) Memory management Unit (MMU) <-------- This is the hardware
   d) Operating System (OS)
   e) c & d

A

9. Which of the following defines the set of instructions loaded into the memory?
   a) Application <-------- An application is composed of tasks
   b) Task
   c) Thread
   d) All of the above

B

10. Which of the following does not use shared memory?
   a) Process <-------- Processes never share memory. The communicate with semaphores
   b) Thread
   c) ISR
   d) Kernel

A

11. Which of the following is inherited from the parent process?
   a) Task
   b) Application
   c) Thread <-------- Threads are launched by tasks
   d) Kernel

C

12. In operating system, each process has its own
   a) address space and global variables
   b) open files
   c) pending alarms, signals and signal handlers <------- A process may never have an ala
   d) A & B &C
   e) A & B Only

E

13. Which of the following scheduling algorithms are based on the assumption that tasks are executed until they are done?
   a) Periodic task
   b) Aperiodic task
   c) Non-preemptive scheduling <-------- As the name implies it cannot be preempted.
   d) Preemptive scheduling

C

Problem 5: (10 points)

Calculate the average IPC of a pipeline machine given the following parameters

- Clock cycle time: 10ns
- Branch instructions generate a 4-cycle stall
- Load Store instructions generate a 3-cycle stall
- Branch instructions occur 12% of the time
- Load/store instructions occur 23% of the time
- All remaining instruction complete in 1 cycle

CPI = 1 + 4*.12 + 3*.23
CPI = 1 + .48 + .69 = 2.17
IPC = 1/CPI = .46

Problem 6: (20 points)

a) If a multiply instructions require 13 cycles and accounts for 10% of the instructions executed in a typical program, what percentage of time does the CPU spend on multiplication if the average CPI of all other instructions is 5 (five)?

13 CPI * 10% + 5 CPI * 90% = 5.8 CPI
Average time spent on multiply instructions: (13*.10)/5.8 = 22.4%

b) Suppose that you could reduce the number of cycles for multiplication to 11 but that the CPU cycle time would need to be increased by 10%. Explain why you should or should not make this modification?

11 CPI * 10% + 5 CPI * 90% = 5.6 CPI
Increasing the cycle time by 10% results in a CPI of 5.6 * 1.1 = 6.16

This is not a good trade off to make as 6.16 is only a 6.2% decrease in performance