Debates/Discussions – Week 5

(1) Which of the components of a process are **shared** across threads in a **multithreaded process**? Explain.
a. Register values b. Heap c. Data d. Stack

(2) Explain why multithreading may allow continued execution even when a system call happens (aka Non-blocking system call)

(3) Consider Figure 4.16. (a) What is the output of Line C and Line P? (b) What happens if **pthread_join()** was not used? (c) What happens if **wait()** was not used? Explain.

(4) Discuss Possible output threads in chapter 4 slides pages 32-35. Explain how case 3 and case 4 can happen.

```c
#include <pthread.h>
#include <stdio.h>

#include <types.h>

int value = 0;
void *runner(void *param); /* the thread */

int main(int argc, char *argv[])
{
pid_t pid;
pthread_t tid;
pthread_attr_t attr;

   pid = fork();

   if (pid == 0) { /* child process */
      pthread_attr_init(&attr);
      pthread_create(&tid,&attr,runner,NULL);
      pthread_join(tid,NULL);
      printf("CHILD: value = %d",value); /* LINE C */
   }
   else if (pid > 0) { /* parent process */
      wait(NULL);
      printf("PARENT: value = %d",value); /* LINE P */
   }
}

void *runner(void *param) {
   value = 5;
   pthread_exit(0);
}
```

Figure 4.16

```
int x = 1; //global variable

void* func(void* p){
   x = x + 1;
   printf("x is %d\n", x);

   // interrupted during printf()

   printf("x is %d\n", x);

   return NULL;
}
```

```
void* func(void* p){
   x = x + 1;
   printf("x is %d\n", x);
   return NULL;
}
```

time

Output:

```
x is 3
x is 2
```

Parent thread **Child thread**

Output:

```
x is 2
x is 2
```

- Is it a possible output for this example ??
  - Hint: translate x = x + 1 into assembly instructions
    - lw $t0, 0($gp)
    - addi $t0, $t0, 1
    - sw $t0, 0($gp)

$t0: data register
$gp: memory address of x
lw: load word (from memory)
sw: store word (to memory)

- Bottom line: We cannot predict the results!
  - We need process (thread) synchronization (Ch. 6)