OS HW #1 Solution

Ch. 2 Ex: 2.3 + 2.4 Pgm Project: 2.1 on 2.2

1 2 2 = 5 total

2.3 1 ms switch cost by OS

Thread A

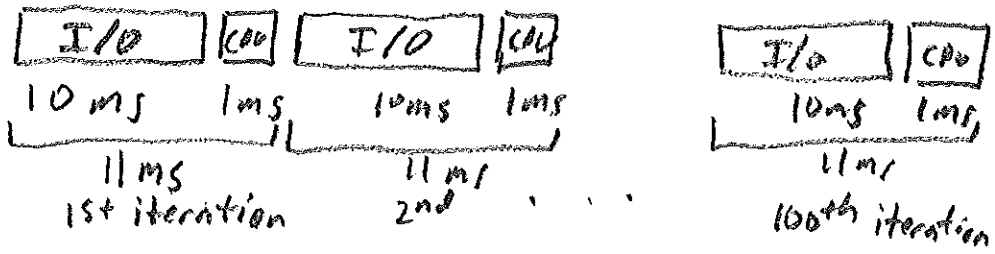
loop 100 times
- I/O - 10 ms
- CPU - 1 ms

Thread B

CPU - 1,000 ms

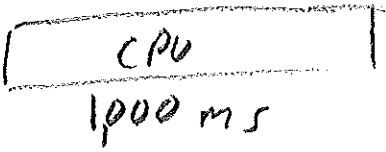
a)

A:



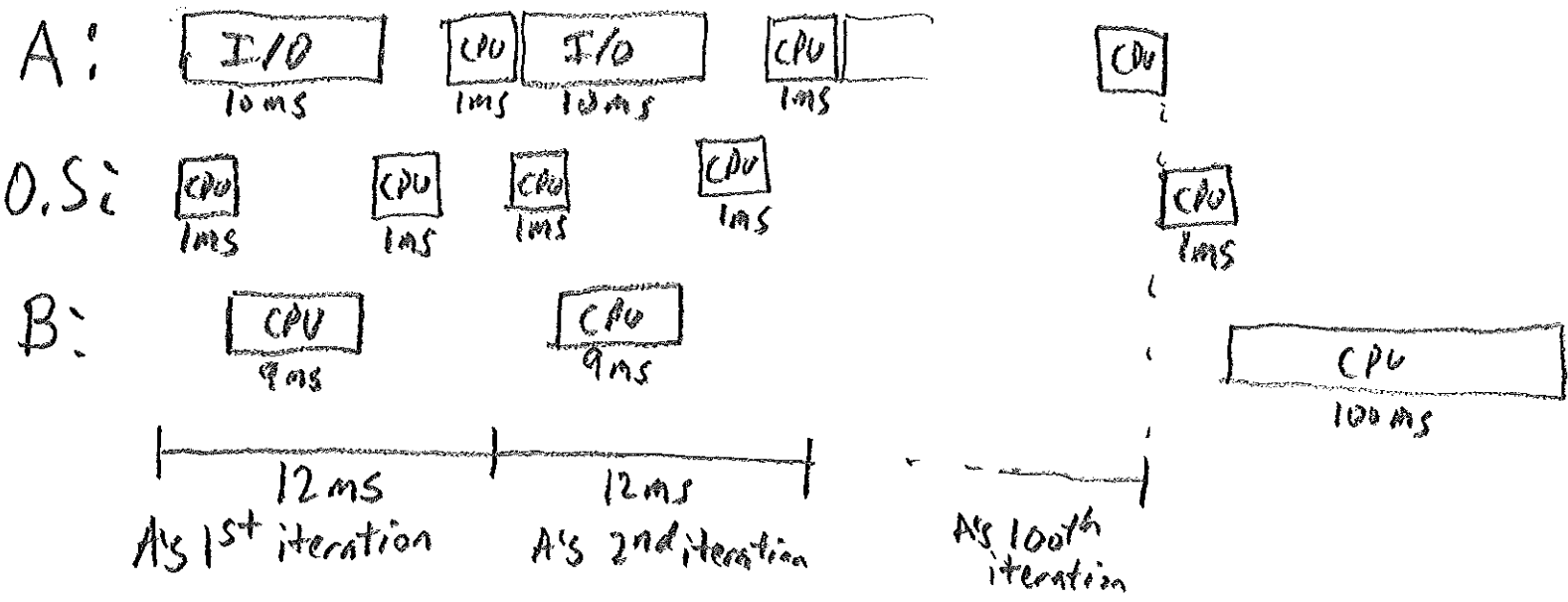
OS: thread 0
switch 0

B: 0
0



total elapsed time = 11 ms × 100 + 1 ms + 1,000 ms = **2,101 ms**

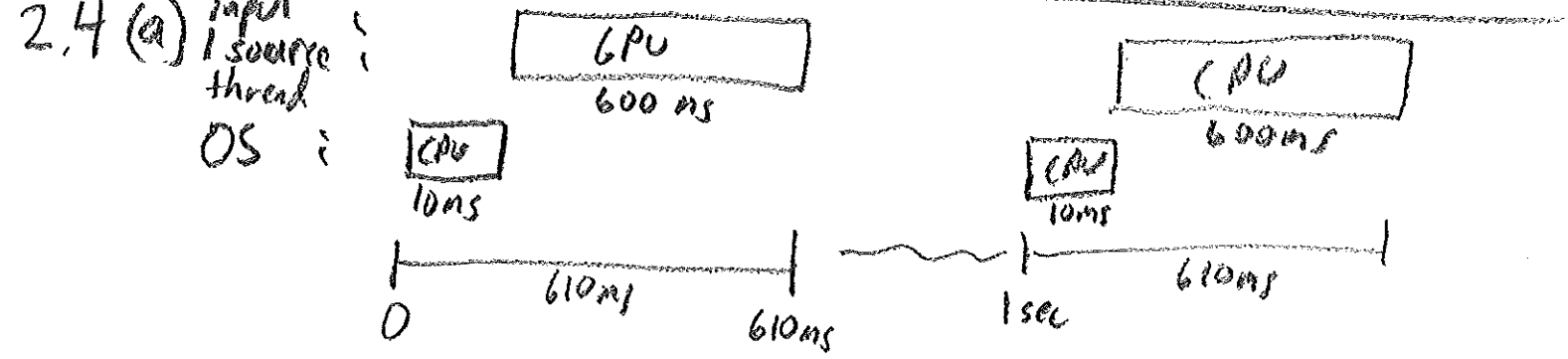
2.3 b)



9ms × 100 = 900ms of
B's CPU burst interleaved

total elapsed time = 12ms × 100 + 1ms + 100ms = (1,301ms)

2.4 (a)

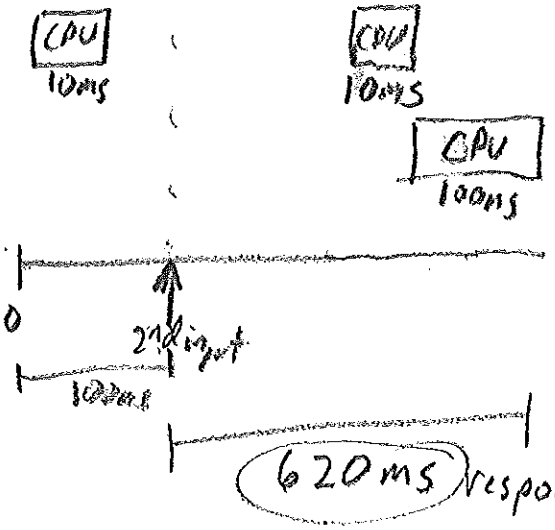


610ms response time

2.4 b) input 1 source threads:

OS:

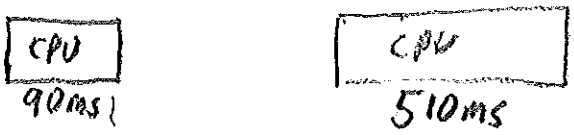
input 2 source thread 8



(610+620)/2 = 615ms average response time for combined input sources

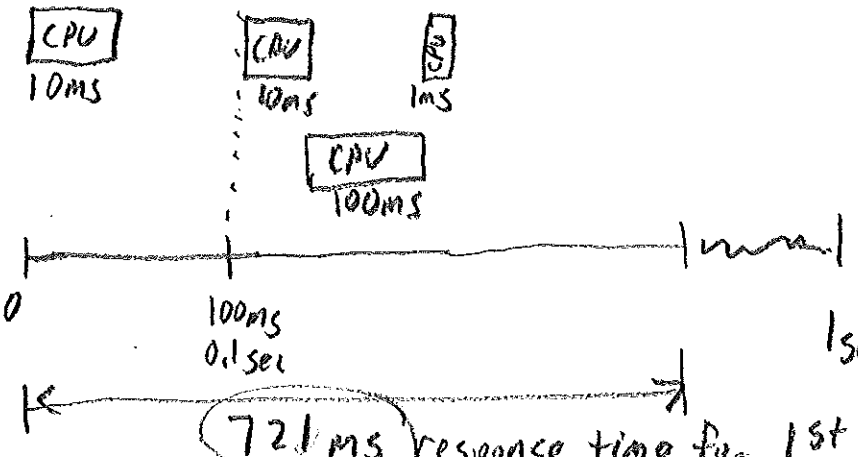
c)

input 1 source threads:



OS:

input 2 source threads:



(720+110)/2 = 415.5ms average combined response time

```
import java.util.Scanner;

public class HW1 {
  public static void main(String args[]){
      String buffer;
      Scanner scan = new Scanner(System.in);
    Thread childThread = new Thread(new Runnable(){
        public void run(){
          System.out.println("Child is started.");
        while (true) {
            sleep(5000);
            System.out.println("Child is done sleeping 5 seconds.");
        } // end while
        }
      });
    childThread.start();
    System.out.print("Hit <Enter> to stop the child\n");
    buffer = scan.nextLine();
    childThread.stop();
    System.out.print("Hit <Enter> to stop the parent\n");
    buffer = scan.nextLine();
  }

  private static void sleep(int milliseconds){
    try{
      Thread.sleep(milliseconds);
    } catch(InterruptedException e){
      // ignore this exception; it won't happen anyhow
    }
  }
}
```

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

static void *child(void *ignored){
  printf("Child process started\n");
  while (1) {
    sleep(5);
    printf("Child is done sleeping 5 seconds.\n");
  } /* end while */
  return NULL;
}

int main(int argc, char *argv[]){
  pthread_t child_thread;
  int code;
  char buffer[100];

  code = pthread_create(&child_thread, NULL, child, NULL);
  if(code){
    fprintf(stderr, "pthread_create failed with code %d\n", code);
    return 0;
  }
  printf("Hit <Enter> key to terminate child process\n");
  scanf("%c",&buffer);

  code = pthread_cancel(child_thread);
  if (code) {
    fprintf(stderr, "pthread_cancel failed with code %d\n", code);
    return 0;
  } /* end if */
  printf("Child was successfully cancelled!\n");
  printf("Hit <Enter> key to terminate parent process\n");
  scanf("%c",&buffer);
  return 0;
}
```