REPORT:
Task 1.1:

1. Header Includes:
- The program includes several header files: ‘kernel/types.h’, ‘kernel/stat.h’, and ‘user/user.h’. These headers likely contain necessary definitions and function prototypes for system calls and data types used in the program.

2. Main Function:
  - The `main` function is the entry point of the program.
  - It accepts command-line arguments (‘argc’ and ‘argv[]’) to determine the number of ticks to sleep.
  - It first checks if the correct number of arguments is provided. If not, it prints a usage message and exits with an error code.
  - It then converts the argument representing the number of ticks to an integer using `atoi`.
  - If the provided number of ticks is non-positive, it prints an error message and exits.
  - Otherwise, it calls the `sleep` function with the specified number of ticks, which suspends execution for that duration.
  - Finally, it exits with a success code.

3. Error Handling:
  - The program performs basic error handling to ensure proper usage and valid input.
  - It prints error messages to standard error (`stderr`) when necessary.

4. Makefile Dependency:
  - We make changes in the MakeFile and add ‘$U/_sleep-cs22btech11012\’ in UPROGS.

5.Screenshot:

```
riscv64-linux-gnu-ld -z max-page-size=4096 -T user/user.ld -o user/_sleep-cs22b
ech11012 user/sleep-cs22btech11012.o user/ulib.o user/usys.o user/printf.o user/
umalloc.o
riscv64-linux-gnu-objdump -S user/_sleep-cs22btech11012 > user/sleep-cs22btech1
012.asm
riscv64-linux-gnu-objdump -t user/_sleep-cs22btech11012 | sed '1,/SYMBOL TABLE/
; s/ .* / /; /^$/d' > user/sleep-cs22btech11012.sym
mkfs/mkfs fs.img README user/_cat user/_echo user/_forktest user/_grep user/_in
t user/_kill user/_ln user/_ls user/_mkdir user/_rm user/_sh user/_stressfs use
/_usertests user/_grind user/_wc user/_zombie user/_sleep-cs22btech11012
nmeta 46 (boot, super, log blocks 30 inode blocks 13, bitmap blocks 1) blocks 1
54 total 2000
balloc: first 778 blocks have been allocated
balloc: write bitmap block at sector 45
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp
3 -nographic -global virtio-mmio.force-legacy=false -drive file=fs.img,if=none,
ormat=raw,id=x0 -device virtio-blk-device,drive=x0,bus=virtio-mmio-bus.0

xv6 kernel is booting

hart 2 starting
hart 1 starting
init: starting sh
$ sleep-cs22btech11012 2
$
```

Task 1.2:

1. Header Includes:
- The program includes several header files: ‘kernel/types.h’, ‘kernel/stat.h’, and ‘user/user.h’. These headers likely contain necessary definitions and function prototypes for system calls and data types used in the program.

2. Main Function:
  - The ‘main’ function is the entry point of the program.
  - It initializes an array ‘p’ to hold file descriptors for a pipe.
  - It then creates a pipe using the ‘pipe’ system call.

3. Forking:
  - The program forks a child process. The child process will handle reading from the pipe, while the parent process will handle writing to the pipe.

4. Child Process:
  - In the child process branch (`fork() == 0`), it reads from the read end of the pipe (`p[0]`) into the `recv_buf` array, expecting 4 bytes.
  - After reading, it closes the read end of the pipe (`p[0]`).
  - It then prints a message indicating the process ID (`getpid()`) and the message received.
  - Next, it writes the message "pong" to the write end of the pipe (`p[1]`), again expecting 4 bytes.

- Finally, it closes the write end of the pipe (`p[1]`) and exits.

5. Parent Process:
   - In the parent process branch, it writes the message "ping" to the write end of the pipe (`p[1]`), expecting 4 bytes.
   - After writing, it closes the write end of the pipe (`p[1]`).
   - It then reads from the read end of the pipe (`p[0]`) into the `recv_buf` array, expecting 4 bytes.
   - After reading, it closes the read end of the pipe (`p[0]`).
   - It prints a message indicating the process ID (`getpid()`) and the message received.
   - Finally, it exits.
6.Screenshot: