This lab investigates calling assembly language procedures from C. Specifically, we look at a function `dump_regs()` which displays the contents of the parameter registers $a0, $a1, $a2, $a3, as well as the return value registers $v0, $v1, without disturbing the code before or after the dump. The function `dump_regs()` does not return a value itself (i.e. its return type is `void`).

1. Get a copy of the program `lab3.c`, and the assembly language module `sample.s` and put it in a subdirectory named `lab3` under `sde/examples`. These files are in `/usr/fac/marc/public_html/code/cs421/lab3` which is a symlink to `/usr/fac/marc/public_html/code/sde/examples/lab3`. Make a Makefile for this program (with any other options you want) which should look something like this

```
PROG = lab3
OBJ = lab3.o sample.o
CFLAGS = -g

include ../make.mk
```

Test the program by compiling

```
sde-make
```

and running

```
sde-run lab3ram
```

2. Check the following things. Note that the first call to `dump_regs()` should show 1025 (0x401) and 1026 (0x402) in registers $v0, $v1, respectively. After the internal function `add()` is called the sum (5) should be in register $v0. Why? Finally, after the call to `printf()`, the last register dump should show 40 (0x28) in register $v0. Why?

3. Once you know the program works correctly note that the function `dump_regs()` has a deficiency in that it doesn’t actually dump all the registers, only $a0, $a1, $a2, $a3, $v0, and $v1.

Assignment Fix the function `dump_regs()` so that it dumps all 32 MIPS registers but restores everything, so that after it returns, nothing has changed (except that the program counter has been advanced). Email me either the path to your solution (which should be in an assembly language source file like `sample.s`) or the text itself (but not an attachment).