PDS1:

We have 32 registers as follows:
The MIPS R2000 CPU has 32 registers. 31 of these are general-purpose registers that can be used in any of the instructions. The last one, denoted register zero, is defined to contain the number zero at all times. Even though any of the registers can theoretically be used for any purpose, MIPS programmers have agreed upon a set of guidelines that specify how each of the regis- ters should be used. Programmers (and compilers) know that as long as they follow these guidelines, their code will work properly with other MIPS code.

Symbolic Name	Number	Usage
zero	0	Constant 0.
at	1	Reserved for the assembler.
v0 - v1	2 - 3	Result Registers.
a0 - a3	4 - 7	Argument Registers $1 \cdots 4$.
t0 - t9	8 - 15, 24 - 25	Temporary Registers $0 \cdots 9$.
s0 - s7	16 - 23	Saved Registers $0 \cdots 7$.
k0 - k1	26 - 27	Kernel Registers $0 \cdots 1$.
gp	28	Global Data Pointer.
sp	29	Stack Pointer.
fp	30	Frame Pointer.
ra	31	Return Address.

In our code, we have used a register bank 'reg' containing 32 '32 bits registers'. Here the register reg[0] contains the initial address of the first instruction. All the instructions are stored in the memory from indices 0 to 16 i.e mem[0] to mem[16]. Also the input unsorted array is stored in the memory from mem[40] to mem[49].

PDS2:

Each instruction is a 32 bit number, stored in memory of size 2^8 = 256. We will use another VEDA memory as data memory of size 2^8 = 256. In total, our VEDA memory

‘MEM’ will be of size 512 X 32 bits. The stack pointer($sp) will point to the topmost row of the data memory.

PDS3:

The layout for R-type instruction is:(RR-ALU)

opcode	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

The opcode field indicates that the instruction is an R-type instruction, and the funct field specifies the specific operation to be performed. The rs, rt, and rd fields specify the source and destination registers for the operation, and the shamt field indicates a shift amount if the instruction is a shift operation.

The layout for I-type instruction is:(RM-ALU)

opcode	rs	rt	immediate
6 bits	5 bits	5 bits	16 bits

The opcode field indicates that the instruction is an I-type instruction. The rs field specifies the source register, and the rt field specifies the destination register. The immediate field contains the immediate value to be used in the operation.

The layout for J-type instruction is:(BNE, BEQ)

opcode	address
6 bits	26 bits

The opcode field indicates that the instruction is a J-type instruction. The address field contains the memory address of the instruction to jump to.

For the purpose of this project we’ll be defining our own opcodes as shown below:

ADD=000000
SUB=000001
AND=000010
OR=000011
SLT=000100
MUL=000101
HLT=111111
LW=001000
SW=001001

ADDI=001010
SUBI=001011
SLTI=001100
BNE=001101
BEQ=001110