Language Description Table
(Brookshear's Appendix A)

16 registers [0..F]   256 memory cells [00..FF]   (8 bits each)
16-bit instructions (4+12)   R, S, T = register numbers   X, Y = values or addresses
3 Machine mode: Base, Stack, and Branch

Mnemonic	**Pattern**	**Op-code**	**Operand**	**Description**
`LEA`	`R{R}, XY`	`1`	`RXY`	LOAD the register R with the bit pattern found in the memory cell whose address is XY. *Example:* `14A3` would cause the contents of the memory cell located at address A3 to be placed in register 4.
`LDR`	`R{R}, XY`	`2`	`RXY`	LOAD the register R with the bit pattern XY. *Example:* `20A3` would cause the value A3 to be placed in register 0.
`STR`	`R{R}, XY`	`3`	`RXY`	STORE the bit pattern found in register R in the memory cell whose address is XY. *Example:* `35B1` would cause the contents of register 5 to be placed in the memory cell whose address is B1.
`MOV`	`R{R}, R{S}`	`4`	`0RS`	MOVE the bit pattern found in register R to register S. *Example:* `40A4` would cause the contents of register A to be copied into register 4.
`ADT`	`R{R}, R{S}, R{T}`	`5`	`RST`	ADD the bit patterns in registers S and T as though they were two's complement representations and leave the result in register R. *Example:* `5726` would cause the binary values in registers 2 and 6 to be added and the sum placed in register 7.
`ADF`	`R{R}, R{S}, R{T}`	`6`	`RST`	ADD the bit patterns in registers S and T as though they represented values in floating-point notation and leave the floating-point result in register R. *Example:* `634E` would cause the values in registers 4 and E to be added as floating-point values and the result to be placed in register 3.
`ORR`	`R{R}, R{S}, R{T}`	`7`	`RST`	OR the bit patterns in registers S and T and place the result in register R. *Example:* `7CB4` would cause the result of OR-ing the contents of registers B and 4 to be placed in register C.
`AND`	`R{R}, R{S}, R{T}`	`8`	`RST`	AND the bit patterns in registers S and T and place the result in register R. *Example:* `8045` would cause the result of AND-ing the contents of registers 4 and 5 to be placed in register 0.
`XOR`	`R{R}, R{S}, R{T}`	`9`	`RST`	XOR the bit patterns in registers S and T and place the result in register R. *Example:* `95F3` would cause the result of XOR-ing the contents of registers F and 3 to be placed in register 5.

ROT	R{R}, 0X	A	R0X	ROTATE the bit pattern in register R one bit to the right X times. Each time place the bit that started at the low-order end at the high-order end. *Example:* `A403` would cause the contents of register 4 to be rotated 3 bits to the right in a circular fashion.
JMP	R{R}, XY	B	RXY	JUMP to the instruction located in the memory cell at address XY if the bit pattern in register R is equal to the bit pattern in register number 0. Otherwise, continue with the normal sequence of execution. (The jump is implemented by copying XY into the program counter during the execute phase.) *Example:* `B43C` would first compare the contents of register 4 with the contents of register 0. If the two were equal, the pattern 3C would be placed in the program counter so that the next instruction executed would be the one located at that memory address. Otherwise, nothing would be done and program execution would continue in its normal sequence.
HLT		C	000	HALT execution. *Example:* `C000` would cause program execution to stop.
PSH	R{R}, R{S}, ...	D	R00	PUSH the bit pattern found in register R to the stack. Multiple registers can be pushed using one instruction. Example: D100 would cause the contents of register 1 to be pushed to the stack Note: This opcode only works in “Stack” mode
POP	R{R}, R{S}, ...	E	R00	POP the bit pattern found in register R from the stack. Multiple registers can be popped using one instruction. Example: E100 would cause the value located at the top of the stack to be placed in register 1. Note: This opcode only works in “Stack” mode Note: POP PC is a special case in “Branch” mode which allows the link register to be placed into the program counter to simulate a return
CALL	FUNC	F	0XY	BRANCH to the instruction located in the memory cell at address XY and save the return address in a dedicated link register. Example: F0AA would cause the pattern AA to be placed in the program counter so that the next instruction executed would be the one located at that memory address. Note: This opcode only works in “Branch” mode