Hints

Addressing Modes:

Immediate	#value	operand = value
Register	R*i*	EA = R*i*
Absolute (Direct)	LOC	EA = LOC
Indirect	(R*i*) (LOC)	EA = [R*i*] EA = [LOC]
Index	X(R*i*)	EA = [R*i*] + X
Base with index	(R*i*, R*j*)	EA = [R*i*] + [R*j*]
Base with index and offset	X(R*i*, R*j*)	EA = [R*i*] + [R*j*] + X
Autoincrement	(R*i*)+	EA = [R*i*]; Increment R*i*
Autodecrement	-(R*i*)	Decrement R*i*; EA = [R*i*] + X

Binary Number Representation Schemes

- Sign and magnitude where the most significant bit being 0 for a positive number and 1 for a negative number.
- One's complement where a negative number is represented by the complement of its positive representation.
- Two's complement where a negative number is derived by adding 1 to its one's complement.

Basic Performance Equation

$$T = N \times S / R$$

- T = Time required to execute a program
- N = Number of machine language instructions
- S = Number of clock cycles per machine instruction
- R = Clock rate in cycles per second (Hz)

Sample Subroutine

Calling Program

```
      …
      Move        PARAM2,-(SP)     ; Place parameters on the stack
      Move        PARAM1,-(SP)
      Call        SUB1
      Move        (SP),RESULT      ; Store result
      Add         #8,SP            ; Restore stack
      …
```

Subroutine

```
SUB1  Move        FP,-(SP)         ; Save frame pointer
      Move        SP,FP            ; Load the frame pointer
      MoveMult    R0-R2,-(SP)      ; Save registers
      Move        8(FP),R0         ; Get first parameter (PARAM1)
      Move        12(FP),R1        ; Get second parameter (PARAM2)
      …                            ; Perform computation
      Move        R2,8(FP)         ; Place the answer on the stack
      MoveMult    (SP)+,R0-R2      ; Restore registers
      Move        (SP)+,FP         ; Restore frame pointer
      Return                       ; Return to calling program
```