The following `MIX` subroutines, for addition and subtraction of numbers having the form (4), show how Algorithms A and N can be expressed as computer programs. The subroutines below are designed to take one input u from symbolic location `ACC`, and the other input v comes from register A upon entrance to the subroutine. The output w appears both in register A and location `ACC`. Thus, a fixed point coding sequence

`LDA A; ADD B; SUB C; STA D` (7)

would correspond to the floating point coding sequence

`LDA A, STA ACC; LDA B, JMP FADD; LDA C, JMP FSUB; STA D.` (8)

Program A (*Addition, subtraction, and normalization*). The following program is a subroutine for Algorithm A, and it is also designed so that the normalization portion can be used by other subroutines that appear later in this section. In this program and in many others throughout this chapter, `OFLO` stands for a subroutine that prints out a message to the effect that `MIX`'s overflow toggle was unexpectedly found to be on. The byte size b is assumed to be a multiple of 4. The normalization routine `NORM` assumes that $\mathrm{rI2} = e$ and $\mathrm{rAX} = f$, where $\mathrm{rA} = 0$ implies $\mathrm{rX} = 0$ and $\mathrm{rI2} < b$.

BYTE	EQU	1(4:4)	Byte size b
EXP	EQU	1:1	Definition of exponent field
FSUB	STA	TEMP	Floating point subtraction subroutine:
	LDAN	TEMP	Change sign of operand.
FADD	STJ	EXITF	Floating point addition subroutine:
	JOV	OFLO	Ensure that overflow is off.
	STA	TEMP	`TEMP` $\leftarrow v$.
	LDX	ACC	$\mathrm{rX} \leftarrow u$.
	CMPA	ACC(EXP)	Steps A1, A2, A3 are combined here:
	JGE	1F	Jump if $e_v \geq e_u$.
	STX	FU(0:4)	`FU` $\leftarrow \pm\, f\, f\, f\, f\, 0$.
	LD2	ACC(EXP)	$\mathrm{rI2} \leftarrow e_w$.
	STA	FV(0:4)	
	LD1N	TEMP(EXP)	$\mathrm{rI1} \leftarrow -e_v$.
	JMP	4F	
1H	STA	FU(0:4)	`FU` $\leftarrow \pm\, f\, f\, f\, f\, 0$ (u, v interchanged).
	LD2	TEMP(EXP)	$\mathrm{rI2} \leftarrow e_w$.
	STX	FV(0:4)	
	LD1N	ACC(EXP)	$\mathrm{rI1} \leftarrow -e_v$.
4H	INC1	0,2	$\mathrm{rI1} \leftarrow e_u - e_v$. (Step A4 unnecessary.)
5H	LDA	FV	*A5. Scale right.*
	ENTX	0	Clear rX.
	SRAX	0,1	Shift right $e_u - e_v$ places.
6H	ADD	FU	*A6. Add.*
	JOV	N4	*A7. Normalize.* Jump if fraction overflow.
	JXZ	NORM	Easy case?
	LD1	FV(0:1)	Check for opposite signs.
	JAP	1F	

The following MIX subroutines, for addition and subtraction of numbers having the form (4), show how Algorithms A and N can be expressed as computer programs. The subroutines below are designed to take one input u from symbolic location ACC, and the other input v comes from register A upon entrance to the subroutine. The output w appears both in register A and location ACC. Thus, a fixed point coding sequence

`LDA A; ADD B; SUB C; STA D` (7)

would correspond to the floating point coding sequence

`LDA A, STA ACC; LDA B, JMP FADD; LDA C, JMP FSUB; STA D.` (8)

Program A (*Addition, subtraction, and normalization*). The following program is a subroutine for Algorithm A, and it is also designed so that the normalization portion can be used by other subroutines that appear later in this section. In this program and in many others throughout this chapter, OFLO stands for a subroutine that prints out a message to the effect that MIX's overflow toggle was unexpectedly found to be on. The byte size b is assumed to be a multiple of 4. The normalization routine NORM assumes that $\mathrm{rI2} = e$ and $\mathrm{rAX} = f$, where $\mathrm{rA} = 0$ implies $\mathrm{rX} = 0$ and $\mathrm{rI2} < b$.

BYTE	EQU	1(4:4)	Byte size b
EXP	EQU	1:1	Definition of exponent field
FSUB	STA	TEMP	Floating point subtraction subroutine:
	LDAN	TEMP	Change sign of operand.
FADD	STJ	EXITF	Floating point addition subroutine:
	JOV	OFLO	Ensure that overflow is off.
	STA	TEMP	TEMP $\leftarrow v$.
	LDX	ACC	$\mathrm{rX} \leftarrow u$.
	CMPA	ACC(EXP)	Steps A1, A2, A3 are combined here:
	JGE	1F	Jump if $e_v \geq e_u$.
	STX	FU(0:4)	FU $\leftarrow \pm f\,f\,f\,f\,0$.
	LD2	ACC(EXP)	$\mathrm{rI2} \leftarrow e_w$.
	STA	FV(0:4)	
	LD1N	TEMP(EXP)	$\mathrm{rI1} \leftarrow -e_v$.
	JMP	4F	
1H	STA	FU(0:4)	FU $\leftarrow \pm f\,f\,f\,f\,0$ (u, v interchanged).
	LD2	TEMP(EXP)	$\mathrm{rI2} \leftarrow e_w$.
	STX	FV(0:4)	
	LD1N	ACC(EXP)	$\mathrm{rI1} \leftarrow -e_v$.
4H	INC1	0,2	$\mathrm{rI1} \leftarrow e_u - e_v$. (Step A4 unnecessary.)
5H	LDA	FV	*A5. Scale right.*
	ENTX	0	Clear rX.
	SRAX	0,1	Shift right $e_u - e_v$ places.
6H	ADD	FU	*A6. Add.*
	JOV	N4	*A7. Normalize.* Jump if fraction overflow.
	JXZ	NORM	Easy case?
	LD1	FV(0:1)	Check for opposite signs.
	JAP	1F	