Frankie

Our processor is named Frankie, in reference to Frankenstein's monster. It is primarily accumulator-based, with parts taken from both stack and load-store architecture.

Its main feature is the presence of two accumulators: a main accumulator ("Mary") and a secondary accumulator ("Shelley"). All commands dealing with immediates are handled by Mary. Many commands also have an option of acting on the two accumulators instead; for example, an "aadd" (accumulator add) command could either add an immediate to Mary, or it could add the value in Shelley to Mary. This relationship between the main and secondary accumulators is fundamental to the architecture's design.

Register File

Mary (main accumulator)

This is the main accumulator. Instructions that interact with immediates will interact directly with this. The value in this register is always treated as a signed number in two's complement.

Shelley (secondary accumulator)

This is the secondary accumulator. It can be used as a backup register. It can also be used to perform operations that involve two accumulators. It generally will not interact with immediates. The value in this register is always treated as a signed number in two's complement.

ra (return address register)

This register stores the address that a procedure call will return from using the jret (jump return) instruction. This is set automatically by the jfnc (jump to function) instruction.

pc (program counter register)

This register stores the address of the current instruction. This is set by various jump instruction.

sp (stack pointer register)

This register stores the address of the top of the stack All operations which manipulate the stack implicitly move the stack pointer; as a result, there is no way to set this directly.

comp (comparison result register)

This register stores the result of a comparison instruction (cequ, cles, or cgre), and can only be set by those instructions.

The Clerval Instruction Set Architecture

*in loving memory of Henry Clerval*

There is only one instruction format.

1 flag bit at the start

This determines whether the instruction will operate on an immediate or on the two accumulators.

If the flag bit is a 0, the command takes an immediate.

If the flag bit is a 1, the command operates on the two accumulators.

Example:

aadd 0 10 adds 10 to Mary.

aadd 1 adds the value in Shelley to Mary.

5 bit op code

This determines which instruction is performed.

8 bit immediate

This is always a signed number in two's complement form and will be implicitly sign-extended if it is less than 8 bits.

Writing an instruction

All instruction names are 4 characters long. Let "code" be the instruction name, "f" be the flag bit, and "i" be the immediate; all instructions would be written out like the following:

code f i

For example, say the user wants to add the immediate value 6 to Mary, the main accumulator. This instruction would be written as follows:

aadd 0 6

In cases where the flag bit has no effect, either a 0 or 1 will suffice. In cases where the immediate has no effect, any value will do.

If the flag bit is left blank, it is assumed to be 1.

If the immediate is left blank, it is assumed to be 0.

The code, of course, cannot be left blank.

Converting an instruction to machine code

To convert an instruction to machine code, the formula is:

(flag bit) + (op code) + (8-bit immediate),

where '+' here is assumed to mean "concatenate."

Consider the instruction from the previous example:

aadd 0 6

The flag bit is 0.

The op code for aadd is 00010.

The 8-bit binary representation of 6 is 00000110.

After concatenating this together, the full machine code instruction is as follows:

(0) + (00010) + (00000110) = 00001000000110

Procedure calling conventions

- When a procedure is called, if the caller requires a backup of the current accumulator values, it is responsible for calling bkac to put them on the stack. It is assumed that the callee is free to overwrite the accumulator values in whatever ways it wishes.

- When a procedure is called, the caller is responsible for backing up ra. The callee is free to overwrite ra; it is assumed to be backed up already.

- The first argument to a procedure goes into Mary. The second argument goes into Shelley. Any additional arguments should be put onto the stack after ra has been backed up.

- After a procedure has concluded, its return value should be put into Mary. If a second return value is needed, it can be put into Shelley. Any additional return values must go onto the stack.

- When a procedure returns, it should no longer have anything remaining on the stack; its stack frame should be completely empty.

Instructions Reference

aput -- "accumulator put" -- op: 00000

Flag bit 0: Puts a value into Mary, overwriting her previous value.

Example:

aput 0 3 # puts 3 into Mary

This command overwrites Mary's value with the number 3.

Flag bit 1: Puts a value into Shelley, overwriting her previous value.

Example:

aput 1 6 # puts 6 into Shelley

This command overwrites Shelley's value with the number 6.

sput -- "stack put" -- op: 00001

Puts an immediate value directly on top of the stack.

The flag bit has no effect on sput.

Example:

sput 0 8 # puts 8 on top of the stack

This command places 8 on top of the memory stack.

aadd -- "accumulator add" -- op: 00010

Flag bit 0: Adds an immediate value to Mary's.

Example:

aput 0 5 # puts 5 into Mary

aadd 0 7 # adds 7 to Mary's current value

After this command is executed, the value in Mary is 12.

Flag bit 1: Adds the value in Shelly to the value in Mary. Shelley is unaffected.

Example:

aput 0 4 # puts 4 into Mary

aput 1 2 # puts 2 into Shelley

aadd 1 # adds Shelley's value to Mary's

After this command is executed, the value in Mary is 6, and the value in Shelley is 2.

asub -- "accumulator sub" -- op: 00011

Flag bit 0: Subtracts an immediate value from Mary's.

Example:

aput 0 5 # puts 5 into Mary

asub 0 7 # subtracts 7 from Mary's current value

After this command is executed, the value in Mary is -2.

Flag bit 1: Subtracts the value in Shelley from the value in Mary. Shelley is unaffected.

Example:

aput 0 4 # puts 4 into Mary

aput 1 2 # puts 2 into Shelley

asub 1 # subtracts Shelley's value from Mary's

After this command is executed, the value in Mary is 2, and the value in Shelley is 2.

spek -- "stack peek" -- op: 00100

Flag bit 0: Copies a value from the stack into Mary. Unlike a true stack peek, spek can traverse down the stack in 16 bit increments.

Example:

sput 0 5 # put 5 on top of the stack

sput 0 7 # put 7 on top of the stack

spek 0 1 # copy the second value on the stack into Mary

After this command is executed, Mary's value is 5. The stack has a 5 on the bottom and a 7 on top.

Flag bit 1: Copies a value from the stack into Shelley. Unlike a true stack peek, spek can traverse down the stack in 16 bit increments.

Example:

sput 0 5 # put 5 on top of the stack

sput 0 7 # put 7 on top of the stack

spek 1 0 # copy the second first on the stack into Shelley

After this command is executed, Shelley's value is 7. The stack has a 5 on the bottom and a 7 on top.

spop -- "stack pop" -- op: 00101

Flag bit 0: Moves the top value of the stack into Mary.

Example:

sput 0 2 # put 2 on top of the stack

spop # move the value on top of the stack into Mary

After this command is executed, Mary's value is 2, and the stack is empty.

Flag bit 1: Moves the top value of the stack into Shelley.

Example:

sput 0 6 # put 6 on top of the stack

spop # move the value on top of the stack into Shelley

After this command is executed, Shelley's value is 6, and the stack is empty.

rpop -- "ra pop" -- op: 00110

Moves the top value of the stack into ra, the return address register. If the top of the stack is not a valid address, a memory exception will likely occur.

Example:

bkra # back up value of ra onto the stack

rpop # move the value on top of the stack into ra

After this command is executed, both ra and the stack are the same as they began.

jimm -- "jump immediate" -- op: 00111

Flag bit 0: Set pc to the address specified by the immediate. If the immediate is not a valid address, a memory exception will likely occur.

Example:

jimm 0 0x0 # jump to the address 0x0

After this command is executed, the value in pc will be 0x0.

Flag bit 1: Add (16\*immediate) to the current pc. This effectively moves (1\*immediate) instructions forward.

Example:

jimm 1 -2 # sets pc to pc-32

After this command is executed, the program will effectively be moved two instructions back.

jacc -- "jump accumulator" -- op: 01000

Flag bit 0: Set pc to the value in Mary. If the value in Mary is not a valid address, a memory exception will likely occur.

Example:

aput 0 0x0 # put 0x0 into Mary

jacc 0 # jump to the address in Mary

After this command is executed, the value in pc will be 0x0.

Flag bit 1: Add (16\*Mary's value) to the current pc. This effectively moves (1\*Mary's value) instructions forward.

Example:

aput 0 -2 # put -2 into Mary

jacc 1 # sets pc to pc-32

After this command is executed, the program will effectively be moved two instructions back.

jcmp -- "jump compare" -- op: 01001

Flag bit 0: Acts exactly like jimm, but only operates if the value in the comp register is 1; otherwise it does nothing.

Example:

aput 0 5 # put 5 into Mary

cles 0 6 # if the value in Mary is less than 6, set the comp register to 1

jcmp 0 0x0 # jump to the address 0x0 if the value in the comp register is 1

After this command is executed, the value in pc will be 0x0.

Flag bit 1: Acts exactly like jimm, but only operates if the value in the comp register is 1; otherwise it does nothing.

Example:

aput 0 5 # put 5 into Mary

cles 0 6 # if the value in Mary is less than 6, set the comp register to 1

jcmp 0 -2 # sets pc to pc-32 if the value in the comp register is 1

After this command is executed, the program will effectively be moved two instructions back.

jret -- "jump return" -- op: 01010

Sets the pc to the value in ra.

The flag bit has no effect on jret.

Example:

jret # sets pc to ra

After this command is executed, the program will continue execution at ra's position.

jfnc -- "jump function" -- op: 01011

Flag bit 0: Acts exactly like jimm, but also sets ra to pc+2 so it can be returned back to with jret.

Example:

jfnc 0x0 # jump to the address 0x0, set ra to pc+2

After this command is executed, the value in pc will be 0x0, and the value in ra will be (starting pc)+2.

Flag bit 1: Acts exactly like jimm, but also sets ra to pc+2 so it can be returned back to with jret.

Example:

jfnc 1 -2 # sets pc to pc-32

After this command is executed, the program will effectively be moved two instructions back, and the value in ra will be (starting pc)+2.

cequ -- "compare equal" -- op: 01100

Flag bit 0: Compares the supplied immediate to the value in Mary. If they are equal, it sets the value in the "comp" register to 1. If they are not, it sets the value in the "comp" register to 0.

Example:

aput 0 6 # set the value in Mary to 6

cequ 0 6 # sets the value in comp to 1 if Mary's value is equal to 6

After this command is executed, the value in Mary will be 6, and the value in comp will be 1.

Flag bit 1: Compares Mary's value to Shelley's value. If they are equal, it sets the value in the "comp" register to 1. If they are not, it sets the value in the "comp" register to 0.

Example:

aput 0 6 # set the value in Mary to 6

aput 1 6 # set the value in Shelley to 6

cequ 1 # sets the value in comp to 1 if Mary's value is equal to Shelley's

After this command is executed, the value in Mary will be 6, the value in Shelley will be 6, and the value in comp will be 1.

cles -- "compare less" -- op: 01101

Flag bit 0: Compares the supplied immediate to the value in Mary. If Mary's value is less than the immediate, it sets the value in the "comp" register to 1. Otherwise it sets the value in the "comp" register to 0.

Example:

aput 0 6 # set the value in Mary to 6

cles 0 7 # sets the value in comp to 1 if Mary's value is less than 6

After this command is executed, the value in Mary will be 6, and the value in comp will be 1.

Flag bit 1: Compares Mary's value to Shelley's value. If Mary's is less than Shelley's, it sets the value in the "comp" register to 1. Otherwise it sets the value in the "comp" register to 0.

Example:

aput 0 6 # set the value in Mary to 6

aput 1 7 # set the value in Shelley to 6

cles 1 # sets the value in comp to 1 if Mary's value is less than Shelley's

After this command is executed, the value in Mary will be 6, the value in Shelley will be 7, and the value in comp will be 1.

cgre -- "compare greater" -- op: 01110

Flag bit 0: Compares the supplied immediate to the value in Mary. If Mary's value is greater than the immediate, it sets the value in the "comp" register to 1. Otherwise it sets the value in the "comp" register to 0.

Example:

aput 0 6 # set the value in Mary to 6

cles 0 5 # sets the value in comp to 1 if Mary's value is less than 5

After this command is executed, the value in Mary will be 6, and the value in comp will be 1.

Flag bit 1: Compares Mary's value to Shelley's value. If Mary's is greater than Shelley's, it sets the value in the "comp" register to 1. Otherwise it sets the value in the "comp" register to 0.

Example:

aput 0 6 # set the value in Mary to 6

aput 1 5 # set the value in Shelley to 6

cles 1 # sets the value in comp to 1 if Mary's value is greater than Shelley's

After this command is executed, the value in Mary will be 6, the value in Shelley will be 5, and the value in comp will be 1.

lorr -- "logical or" -- op: 01111

Flag bit 0: Performs a bitwise "or" between the value in Mary and the supplied immediate, and puts the result in Mary. If necessary, this instruction zero-extends the smaller value.

Example:

aput 0 4 # sets the value in Mary to 4, or 0b100

lorr 0 2 # performs bitwise "or" on the value in Mary and 2, or 0b010

After this command is executed, the value in Mary will be 0b110, or 6.

Flag bit 1: Performs a bitwise "or" between the value in Mary and the value in Shelley, and puts the result in Mary. If necessary, this instruction zero-extends the smaller value.

Example:

aput 0 4 # sets the value in Mary to 4, or 0b100

aput 1 1 # sets the value in Shelley to 1, or 0b001

lorr 1 # performs bitwise "or" on the value in Mary and the value in Shelley

After this command is executed, the value in Mary will be 0b101, or 5.

land -- "logical and" -- op: 10000

Flag bit 0: Performs a bitwise "and" between the value in Mary and the supplied immediate, and puts the result in Mary. If necessary, this instruction zero-extends the smaller value.

Example:

aput 0 4 # sets the value in Mary to 4, or 0b100

land 0 2 # performs bitwise "and" on the value in Mary and 2, or 0b010

After this command is executed, the value in Mary will be 0b000, or 0.

Flag bit 1: Performs a bitwise "and" between the value in Mary and the value in Shelley, and puts the result in Mary. If necessary, this instruction zero-extends the smaller value.

Example:

aput 0 4 # sets the value in Mary to 4, or 0b100

aput 1 1 # sets the value in Shelley to 1, or 0b001

land 1 # performs bitwise "and" on the value in Mary and the value in Shelley

After this command is executed, the value in Mary will be 0b000, or 0.

shfl -- "shift left" -- op: 10001

Flag bit 0: Performs a bitwise left shift on the value in Mary by the number of bits specified by the immediate. This instruction zero-extends from the right.

Example:

aput 0 2 # sets the value in Mary to 2, or 0b010

shfl 0 1 # shifts the value in Mary left by 1 bit

After this command is executed, the value in Mary will be 0b100, or 4.

Flag bit 1: Performs a bitwise left shift on the value in Mary by the number of bits specified in Shelley. This instruction zero-extends from the right.

Example:

aput 0 1 # sets the value in Mary to 1, or 0b001

aput 1 2 # sets the value in Shelley to 2

shfl 1 # shifts the value in Mary left by the number of bits specified by Shelley

After this command is executed, the value in Mary will be 0b100, or 4.

shfr -- "shift right" -- op: 10010

Flag bit 0: Performs a bitwise right shift on the value in Mary by the number of bits specified by the immediate. This instruction sign extends from the left.

Example:

aput 0 2 # sets the value in Mary to 2, or 0b010

shfr 0 1 # shifts the value in Mary right by 1 bit

After this command is executed, the value in Mary will be 0b001, or 1.

Flag bit 1: Performs a bitwise right shift on the value in Mary by the number of bits specified in Shelley. This instruction sign extends from the left.

Example:

aput 0 4 # sets the value in Mary to 1, or 0b100

aput 1 1 # sets the value in Shelley to 1

shfr 1 # shifts the value in Mary right by the number of bits specified by Shelley

After this command is executed, the value in Mary will be 0b010, or 2.

load -- "load from memory" -- op: 10011

Flag bit 0: Loads the value from memory at the address specified in the immediate and copies it into Mary.

Example:

load 0 0x0 # loads the value at address 0x0 in memory and copies it into Mary.

After this command is executed, the value in Mary will be the value at the address 0x0 in memory.

Flag bit 1: Loads the value from memory at the address stored in Mary and copies it into Mary.

Example:

aput 0 0x0 # sets Mary's value to 0x0

load 1 # loads the value at the address in Mary from memory and copies it into Mary

After this command is executed, the value in Mary will be the value at the address 0x0 in memory.

stor -- "store in memory" -- op: 10100

Flag bit 0: Stores the value in Mary into memory at the address specified by the immediate.

Example:

aput 0 2 # sets Mary's value to 2

stor 0 0x0 # stores the value in Mary at the address 0x0 in memory

After this command is executed, the value in Mary will be 2, and the value at 0x0 in memory will also be 2.

Flag bit 1: Stores the value in Mary into memory at the address specified by Shelley.

Example:

aput 0 2 # sets Mary's value to 2

aput 1 0x0 # sets the value in Shelley to 0x0

stor 1 # stores the value in Mary at the address specified by the value in Shelley

After this command is executed, the value in Mary will be 2, the value in Shelley will be 0x0, and the value at 0x0 in memory will be 2.

bkac -- "back up accumulator" -- op: 10101

Flag bit 0: Copies the value in Mary and places it on top of the stack.

Example:

aput 0 2 # sets Mary's value to 2

bkac 0 # copies the value in Mary onto the stack

After this command is executed, the value in Mary will be 2, and the value at the top of the stack will also be 2.

Flag bit 1: Copies the value in Shelley and places it on top of the stack.

Example:

aput 1 3 # sets Shelley's value to 3

bkac 1 # copies the value in Shelley onto the stack

After this command is executed, the value in Shelley will be 3, and the value at the top of the stack will also be 3.

bkra -- "back up return address" -- op: 10110

Copies the value in ra and places it on top of the stack.

The flag bit has no effect on bkra.

Example:

bkra # copies ra onto the stack

After this command is executed, the value on top of the stack will be whatever ra started as.

swap -- "swap the accumulators" -- op: 10111

Swaps the value in Mary with the value in Shelley.

The flag bit has no effect on swap.

Example:

aput 0 5 # sets the value in Mary to 5

aput 1 8 # sets the value in Shelley to 8

swap # swaps the values in Mary and Shelley

After this command is executed, the value in Mary will be 8, and the value in Shelley will be 5.