Team:

- Jordan Cottle

- Michael Russell

- Harshil Suthar

Notes:

- Mathematical operations only performed on floating point values

- DataTypes:

- Floating point values

- Register Addresses

- Memory Addresses

-Memory:

- 1024 Addressable memory locations (32 bits each)

- 16 Addressable registers (32 bits each)

-Speeds:

- Clock cycle: 100ns

- Single ALU operation: 200ns

- Register Read/Write: 100n

- Memory Read/Write: 300ns

- Condition Codes: Z N V E (CAPS = Used)

- Z: Zero flag - set when operation results in a 0

- N: Negative flag - set when operation results in a negative value

- V: Overflow flag - set when operation results in an overflowed operation

- E: Error flag - set when an operations results in an error

- Conventions:

- Rd signifies a destination register

- Ri, Rj signify source registers

- # precedes immediate values

- Encoding format:

- Instructions encoded with 5 bits

- Registers addressed with 5 bits

- Branching labels use all remaining bits

Instruction Mnemonic and Operands Operation Cycles in ALU

-------------------------------------------------------------------------------------

Set Set Rd, #FPvalue Rd ← FPvalue 1

* description: Takes the immediate floating-point value and sets it into the Register specified by Rd
* syntax: SET Rd, #FPValue
* opcode: 00000
* condition codes used: Z N \_ \_

Load Load Rd, Ri Rd ← M[Ri] 1

* description: Takes a value from the specified address in memory (stored in register Ri) and stores it in register specified by Rd
* syntax: LOAD Rd, Ri
* opcode: 00001
* condition codes used: Z N \_ \_

Store Store Rd, Ri M[Rd] ← Ri 1

* description: Stores a value located in register Ri into the memory adressed by the value in register Rd
* syntax: STORE Rd, Ri
* opcode: 00010
* condition codes used: Z N \_ \_

Move Move Rd, Ri Rd ← Ri 1

* description: Moves a value from register Ri into register Rd
* syntax: MOVE Rd, Ri
* opcode: 00011
* condition codes used: Z N \_ \_

Add Fadd Rd, Ri, Rj Rd ← Ri + Rj 3

* description: Adds the floating point values in registers Ri and Rj together and stores the result in Rd
* syntax: ADD Rd, Ri Rj
* opcode: 00100
* condition codes used: Z N V \_

Subtract Fsub Rd, Ri, Rj Rd ← Ri – Rj 3

* description: Subtracts the floating point value in register Rj from the value in Ri and stores it in register Rd
* syntax: SUB Rd, Ri Rj
* opcode: 00101
* condition codes used: Z N V \_

Negate Fneg Rd, Ri Rd ← -Ri 1

* description: Multiplies value in register Ri by -1 and stores result in register Rd
* syntax: NEG Rd, Ri
* opcode: 00110
* condition codes used: Z N \_ \_

Multiply Fmul Rd, Ri, Rj ← Ri \* Rj 5

* description: Multiplies value in registers Ri and Rj together and stores result in register Rd
* syntax: MUL Rd, Ri Rj
* opcode: 00111
* condition codes used: Z N V \_

Divide Fdiv Rd, Ri, Rj ← Ri / Rj 8

* description: Divides value in Ri by value in Rj and stores result in register Rd
* syntax: DIV Rd, Ri Rj
* opcode: 01000
* condition codes used: Z N V E

Floor Floor Rd, Ri Rd ← roundDown(Ri) 1

* description: Calculates the floor of the value in register Ri by rounding it down to the nearest integer value and stores it in register Rd
* syntax: FLOOR Rd Ri
* opcode: 01001
* condition codes used: Z N \_ \_

Ceiling Ceil Rd, Ri Rd ← roundUp(Ri)

* description: Calculates the ceiling of the value in register Ri by rounding it up to the nearest integer value and stores it in register Rd
* syntax: CEIL Rd Ri
* opcode: 01010
* condition codes used: Z N \_ \_

Round Round Rd, Ri Rd ← round(Ri) 1

* description: Rounds the value in register Ri to the neartest integer and stores the result in register Rd
* syntax: ROUND Rd, Ri
* opcode: 01011
* condition codes used: Z N \_ \_

Absolute Value Fabs Rd, Ri Rd ← abs(Ri) 1

* description: Calculates the absolute value of the value in register Ri and stores the result in register Rd
* syntax: ABS Rd, Ri
* opcode: 01100
* condition codes used: Z N \_ \_

Minimum Min Rd, Ri, Rj Rd ← min(Ri, Rj 1

* description: Finds the smaller of the values stored in registers Ri and Rj and stores the result in register Rd
* syntax: MIN Rd, Ri Rj
* opcode: 01101
* condition codes used: Z N \_ \_

Maximum Max Rd, Ri, Rj Rd ← max(Ri, Rj) 1

* description: Finds the larger of the values stored in registers Ri and Rj and stores the result in register Rd
* syntax: MAX Rd, Ri Rj
* opcode: 01110
* condition codes used: Z N \_ \_

Power Pow Rd, Ri, #int\_value Rd ← Ri^integer\_value 6

* description: Raises the value in register Ri to the power specified by the immediate integer value and stores the result in register Rd
* syntax: POW Rd, Ri #int\_value
* opcode: 01111
* condition codes used: Z N V \_

Exponent Exp Rd, Ri Rd ← e^Ri 8

* description: Calculates the value of e raised to the value stored in register Ri and stores the result in register Rd
* syntax: EEXP Rd, Ri
* opcode: 10000
* condition codes used: \_ \_ V \_

Square Root Sqrt Rd, Ri Rd ← sqrt(Ri) 8

* description: Calculates the square root of the value stored in register Ri and stores the result in register Rd
* syntax: SQRT Rd, Ri
* opcode: 10001
* condition codes used: Z \_ \_ E

Branch (Uncond.) B Ri PC ← M[Ri] 1

* description: Moves program counter to the stored value in the memory addressed by the value in register Ri
* syntax: GOTO Ri
* opcode: 11011
* condition codes used: \_ \_ \_ E

Branch Zero BZ Ri, LABEL If (Ri == 0) PC ← LABEL (line) 3

* description: Moves to program counter to the specified label if the value in register Ri is 0
* syntax: GOTOZ Ri LABEL
* opcode: 11000
* condition codes used: \_ \_ \_ E

Branch Negative BN Ri, LABEL If (Ri < 0) PC ← LABEL (line) 3

* description: Moves the program counter to the specified label if the value in register Ri is negative
* syntax: GOTON Ri LABEL
* opcode: 11001
* condition codes used: \_ \_ \_ E

No-op Nop No operation 1

* description: Does nothing for one cycle
* syntax: PASS
* opcode: 11111
* condition codes used: \_ \_ \_ \_

Halt Halt Stop Program -

* description: Stops executing of the program
* syntax: STOP
* opcode: 10101
* condition codes used: \_ \_ \_ \_