RSSB SIMULATOR DOCUMENTATION

RYLEY JEWSBURY

1. INSTRUCTION SET

All the available scripts.
Movement:

- INIT A
- MOV A, B
- MOVN A, B
- NEG A
- SWAP A, B
- LOAD A, [B]
- LOAD A, [B,C]
- STR A, [B]
- STR A, [B,C]
- PUSH A
- POP A

Arithmetic:

- ADD A, B, C
- ADD A, B
- SUB A, B, C
- SUB A, B
- SUBP A, B
- SUBN A, B

Control:

- IFLT A, B
- IFGT A, B
- ELSE
- END
- B label
- BL label
- BX A
- BXL A
- HALT
- NOP

2. DESCRIPTIONS

1. **Data Movement Scripts.**

INIT A. :
Sets A and ACC to 0

MOV A, B. :
moves the value from B into A
if &A = &B, replace with NOP

MOVN A, B. :
moves the negative value -B into A
note: A cannot be the same register as B. use NEG instead

NEG A. :
negates the value in A

SWAP A, B. :
moves the value from A into B, and the value from B into A
if &A = &B, replace with NOP

LOAD A, [B]. :
Loads the data from the address stored in B into A

LOAD A, [B,C]. :
Loads the data from the address stored in B+C into A
assumes that B is positive and B+C is positive

STR A, [B]. :
Stores the data from A into the address stored in B

STR A, [B,C]. :
Stores the data from A into the address stored in B+C

PUSH A. :
Stores the data from A on the stack

POP A. :
Stores the data from the top of the stack in A

2. **Arithmetic Scripts.**

ADD A, B, C. :
Stores the result B+C in A

ADD A, B. :
Stores the result A+B in A

SUB A, B, C. :
Stores the result B-C in A

SUB A, B. :

Stores the result A-B in A. **note:** you would think this script would be really short, but the skipping of RSSB ruins it, so some shorter scripts are provided by SUBP and SUBN

SUBP A, B. :

Stores the result A-B in A
requires that B is positive or zero
if B is negative, NOP

SUBN A, B. :

Stores the result A-B in A
requires that B is negative
if B is positive or zero, NOP

3. **Flow Control Scripts.**

IFLT A, B. :

continues execution until ELSE if A $<$ B
jumps to ELSE otherwise (when A $\geq$ B)

IFGT A, B. :

continues execution until ELSE if A $>$ B
jumps to ELSE otherwise (when A $\leq$ B)
note: testing if two numbers are equal can be done by using both IFLT and IFGT

ELSE. :

separates conditional blocks
note: must always be included, even if the else block is empty

END. :

ends a conditional block
note: must always be included after ELSE

B label. :

Jumps a number of steps. distance to labels must be pre-computed by the compiler

BL label. :

Jumps a number of steps. distance to labels must be pre-computed by the compiler
Updates the Link Register (LR)

BX A. :

Jumps to the address stored in A
note: Typically used as (BX LR) to return from a function

BXL A. :

Jumps to the address stored in A
Updates the Link Register

HALT. :

Stops execution
note: due to the simple hardware, stopping just busy-loops on addresses 0 to 2