State = (Instruction number, acc={0,1})

LOAD var:

Go to a spot on the tape, change the acc based on the symbol on the tape

LOAD imm:

Set the acc

STORE var:

Go to a spot on the tape, change the symbol on the tape based on the acc

DC var imm:

Go to a spot on the tape, set the symbol on the tape

NOT:

Change the acc based on its current value

[AND|OR|XOR] var:

Go to a spot on the tape, change the acc based on its current value and the symbol on the tape

BRANCH instr0 instr1:

Change the instruction number based on the acc

State = (Instruction number, acc={0,1,2,3})

Symbols = {0,1,0’,1’,\_,*vars*}

LOAD[NEXT][BIG] var (acc|temp):

Load the first unread bit of var, mark as read if LOADNEXT

If out of bounds, jump to the next *oob* label

STORE[NEXT][BIG] var (acc|temp)

Write the first unread bit of var, mark as read if STORENEXT

If out of bounds, jump to the next *oob* label

UNREAD var

Mark every bit in a variable as unread

MAP map

Change the acc and/or the temp based on a map function

{0,1,2,3}x{0,1} -> {0,1,2,3} if after a LOAD[NEXT][BIG],

{0,1,2,3} -> {0,1,2,3}x{0,1} if before a STORE[NEXT][BIG],

{0,1,2,3}x{0,1} -> {0,1,2,3}x{0,1} if directly between a LOAD and STORE[NEXT] or a LOADBIG and STORE[NEXT]BIG

{0,1,2,3} -> {0,1,2,3} otherwise

BRANCH (instr | instr0 instr1 instr2 instr3)

Change the instruction number based on the acc

LOADI imm (acc|temp)

ADD vard = var0 + var1

*start:*

LOADNEXT var0 temp

MAP {0x0->0, 0x1->1, 1x0->1, 1x1->2}

LOADNEXT var1 temp

MAP {0x0->0, 0x1->1, 1x0->1, 1x1->2, 2x0->2, 2x1->3}

MAP {0->0x0, 1->0x1, 2->1x0, 3->1x1}

STORENEXT vard temp

JUMP start

*oob:*

UNREAD var0

UNREAD var1

UNREAD vard

SLL vard = var0<<1 + imm

LOADNEXTBIG var0 temp

*start:*

LOADNEXTBIG var0 acc

STORENEXTBIG vard acc

JUMP start

*oob:*

LOADI 0 temp

STORENEXTBIG vard temp

UNREAD var0

UNREAD vard

SRL vard = var0>>1 + imm

LOADNEXT var0 temp

*start:*

LOADNEXT var0 acc

STORENEXT vard acc

JUMP start

*oob:*

LOADI 0 temp

STORENEXT vard temp

UNREAD var0

UNREAD vard

MOVE vard = var0

*start:*

LOADNEXT var0 acc

STORENEXT vard acc

JUMP start

*oob:*

UNREAD var0

UNREAD vard

ADD vard += var0

*start:*

LOADNEXT var0 temp

MAP {0x0->0, 0x1->1, 1x0->1, 1x1->2}

LOAD vard temp

MAP {0x0->0, 0x1->1, 1x0->1, 1x1->2, 2x0->2, 2x1->3}

MAP {0->0x0, 1->0x1, 2->1x0, 3->1x1}

STORENEXT vard temp

JUMP start

*oob:*

UNREAD var0

UNREAD vard

MULT vard += var0 \* var1 (var0 is destroyed)

*start:*

LOADNEXT var0 acc

BRANCH shift add null null

*add:*

ADD vard += var1

*shift:*

SLL var1 <<= 1

JUMP start

*oob:*

UNREAD var0

SLT (var0 < var1)

LOADI 0 acc

*start:*

LOADNEXTBIG var0 acc

LOADNEXTBIG var1 temp

MAP {0x0->0, 0x1->1, 1x0->**3**, 1x1->0}

BRANCH start oob null oob

*oob:*

UNREAD var0

UNREAD var1

SLL vard<<=1 + imm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acc \ symbol | X | 0 | 1 | *else* |
| **0** | X, R, **imm** | 0, R, **0** | 0, R, **1** | *next instruction* |
| **1** | 1, R, **0** | 1, R, **1** |

SRL vard>>=1 + imm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acc \ symbol | Y | 0 | 1 | *else* |
| **0** | Y, R, **imm** | 0, L, **0** | 0, L, **1** | *next instruction* |
| **1** | 1, L, **0** | 1, L, **1** |

SUB vard = var0 - var1

*start:*

LOADNEXT var0 temp

MAP {0x0->0, 0x1->1, **3**x0->**3**, **3**x1->0}

LOADNEXT var1 temp

MAP {0x0->0, 0x1->**3**, 1x0->1, 1x1->0, **3**x0->3, **3**x1->**2**}

MAP {0->0x0, 1->0x1, **2**->1x0, **3**->1x1}

STORENEXT vard temp

JUMP start

*oob:*

UNREAD var0

UNREAD var1

UNREAD vard

SUB vard -= var0

*start:*

LOADNEXT var0 temp

MAP {0x0->0, 0x1->1, **3**x0->**3**, **3**x1->0}

LOAD vard temp

MAP {0x0->0, 0x1->**3**, 1x0->1, 1x1->0, **3**x0->3, **3**x1->**2**}

MAP {0->0x0, 1->0x1, **2**->1x0, **3**->1x1}

STORENEXT vard temp

JUMP start

*oob:*

UNREAD var0

UNREAD vard

DIV varq R varr = var0 / var1

ZERO varr

*start:*

SLL varr <<= 1

LOADNEXTBIG var0 acc

STORE varr acc

SLT varr < var1

MAP {0->0x1, 1->1x0, **3**->0x1}

STORENEXTBIG varq temp

BRANCH sub start null null

*sub:*

SUB varr -= var1

JUMP start

*oob:*

UNREAD var0

UNREAD varq

RECP varq R varr = 1 / var1

ZERO varr

*start:*

SLL varr<<=1 + 1

SLT varr < var1

MAP {0->0x1, 1->1x0, **3**->0x1}

STORENEXTBIG varq temp

BRANCH sub start null null

*sub:*

SUB varr -= var1

JUMP start

*oob:*

UNREAD varq

BOL vard = map(var0 x var1)

*start:*

LOADNEXT var0 acc

LOADNEXT var1 temp

MAP map

STORENEXT var­d acc

*oob:*

UNREAD var0

UNREAD var1

UNREAD vard

BOL vard = map(var0 x vard)

*start:*

LOADNEXT var0 acc

LOAD vard temp

MAP map

STORENEXT var­d acc

*oob:*

UNREAD var0

UNREAD vard

*ALL*

|  |  |  |  |
| --- | --- | --- | --- |
| State \ symbol | X | \_ | *else* |
| **searchL** | X, R, **foundR** | \_, R, **searchR** | *else,* L, **searchL** |
| **searchR** | \_, L, **searchL** | *else,* R, **searchR** |

NOT vard

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| State \ symbol | X | 0 | 1 | Y |
|  | X, R, | 1, R, | 0, R, | *next instruction* |

ZERO vard

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| State \ symbol | X | 0 | 1 | Y |
|  | X, R, | 0, R, | 0, R, | *next instruction* |

ADDI vard imm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| State \ symbol | X | 0 | 1 | Y |
| **0** | X, R, **imm** | 0, R, **0** | 1, R, **0** | *next instruction* |
| **1** | 1, R, **0** | 0, R, **1** |
| **2** | 0, R, **1** | 1, R, **1** |
| **3** | 1, R, **1** | 0, R, **2** |

SUBI vard imm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| State \ symbol | X | 0 | 1 | Y |
| **0** | X, R, **imm** | 0, R, **0** | 1, R, **0** | *next instruction* |
| **1** | 1, R, **1** | 0, R, **0** |
| **2** | 0, R, **1** | 1, R, **1** |
| **3** | 1, R, **2** | 0, R, **1** |

COMP vard

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| State \ symbol | X | 0 | 1 | Y |
| **0** | X, R, **1** | 1, R, **0** | 0, R, **0** | *next instruction* |
| **1** | 0, R, **1** | 1, R, **0** |

SEZ vard

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| State \ symbol | X | 0 | 1 | Y |
|  | X, R, | 0, R, | 1, R, **0**, *next instruction* | Y, R, **1**, *next instruction* |

[LOAD|STORE][NEXT] X temp

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| State \ symbol | X | *0|1* | *0’|1’* | Y |
| **foundR** | *not possible* | *next instruction* | *else,* R, **foundR** | *oob* |

[LOAD|STORE][NEXT]BIG X temp

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| State \ symbol | Y | *0|1* | *0’|1’* | X |
| **foundR** | *not possible* | *next instruction* | else, L, **foundR** | *oob* |

UNREAD X

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| State \ symbol | X | *0|1* | *0’|1’* | *else* |
| **foundR** | *not possible* | *0|1,* R, **foundR** | | *next instruction* |

MAP, BRANCH, and LOADI do not correspond to actual states, they just affect the state transitions and symbols written on other lines

Pi Program

*start:*

RECP V = 1/N

LOAD S acc

BRANCH add null null null

COMP V

*add:*

ADD P += V

ADDI N += 2

BRANCH null end null null

NOT S

JUMP start

*end:*

END