

ToyPacer Assembly Language Specification

ToyPacer Programming Model

The ToyPacer architecture has the following visible registers:

- four 16-bit general purpose registers (R0-3)
- one special purpose Data Segment Register (DSR)

It also has four visible 10-bit “I/O ports”:

- three special purpose Interrupt Service Routine Address registers (ISRADDR0-2)
- one actual I/O port (OUTPORT) 

Finally, it has an Interrupt Enable Bit (IEB) that controls whether or not interrupts are processed.

ToyPacer is a load-store architecture, meaning that only load and store instructions access memory. Computation instructions operate only on values in general purpose registers.

Assembly Language Instructions

Arithmetic Instructions

Addition

ADD RD, RS

Put the sum of registers RD and RS into register RD.

Addition immediate

ADDI RD, IMM

Put the sum of register RD and the sign-extended immediate IMM into register RD.

Subtraction

SUB RD, RS

Put the difference of registers RD and RS into register RD.

Subtraction immediate

SUBI RD, IMM

Put the difference of register RD and the sign-extended immediate IMM into register RD.

Branch Instructions

Branch if greater than or equal

BGE R1, R2, LABEL

Branch to the instruction at LABEL if register R1 is greater than or equal to register R2.

Load and Store Instructions

Load word

LOAD RD, ADDRESS

Load the 16-bit word at address DSR || ADDRESS into register RD.

Set data segment register

SETSEG IMM

Put the 6-bit immediate IMM in DSR.

Store word

STORE RS, ADDRESS

Store the 16-bit word in register RS at address DSR || ADDRESS.

Data Movement Instructions

Move

MOVE RD, RS

Copy register RS to register RD.

Interrupt and Input/Output Instructions

Assert

ASSERT PORT, IMM

Write the 10-bit immediate IMM to port PORT.

Mask interrupt

MASKI

Toggle the IEB.

Return from interrupt

RFI

Return to the interrupted instruction and set the IEB.

Assembly Language Implementation of Sample Pseudocode

```
INIT: MASKI
      SETSEG      0
      ASSERT     ISRADDR0, TINTR
      ASSERT     ISRADDR1, VINTR
      ASSERT     ISRADDR2, AINTR
      SUB        R0, R0
      STORE R0, VTIME
      STORE R0, ATIME
      STORE R0, PTIME
      MASKI
```

```
LOOP: LOAD      R0, VTIME
      ADDI      R0, VDELAY
      LOAD      R1, PTIME
      ADDI      R1, PDELAY
      BGE      R0, R1, CHK
      MOVE     R0, R1
CHK:  LOAD      R1, CTIME
      BGE      R1, R0, LOOP
      ASSERT   OUTPORT, PEVENT
      STORE R1, PTIME
      BGE      R0, R0, LOOP
```

```
TINTR:
      STORE R0, TSAVE
      LOAD   R0, CTIME
      ADDI  R0, 1
      STORE R0, CTIME
      LOAD  R0, TSAVE
      RFI
```

```
VINTR:
      STORE R0, VSAVE
      LOAD   R0, CTIME
      STORE R0, VTIME
      LOAD  R0, VSAVE
      RFI
```

```
AINTR:
      STORE R0, ASAVE
      LOAD   R0, CTIME
      STORE R0, ATIME
      LOAD  R0, ASAVE
      RFI
```

```
VTIME:  0
ATIME:  0
PTIME:  0
CTIME:  0
VSAVE:  0
ASAVE:  0
TSAVE:  0
```