Appendix A - Instruction Set Manual

Note: This document takes time to write and will not be done before the deadline of the functional specifications. You may refer to the program manager for further information in the meantime.

Note for all immediate instruction: Since the difference between a register and an immediate value can easily be made, the mnemonics with immediate values (I type) can be written like their register (R type) counterpart. Example `addi ra 5` can be written `add ra 5`.

ADD - Addition

Description

Adds the content of two registers without carry and stores the result in the destination register.

Syntax

`add [rd] rs rt`

Operands

- `rd`: Optional destination register, defaults to `rs`
- `rs`: First source register
- `rt`: Second source register

Operation

`rd <- rs + rt`

Machine code

`0000000? ???????? ?SSSSSSS SSSDDDDD`

Restrictions

`rd` cannot be `sp` or `ip`. Same condition applies for `rs` if no destination register is specified.

Example

```
add rc ra rb // rc = ra + rb
add rz sp // rz += sp
```

SUB - Subtraction

Description

Subtracts the content of the second register from the content of the first one without carry and stores the result in the destination register.

Syntax

`sub [rd] rs rt`

Operands

- `rd`: Optional destination register, defaults to `rs`
- `rs`: First source register
- `rt`: Second source register

Operation

`rd <- rs - rt`

Machine code

`0000000? ???????? ?SSSSSSS SSSDDDDD`

Restrictions

`rd` cannot be `sp` or `ip`. Same condition applies for `rs` if no destination register is specified.

Example

```
sub rc ra rb // rc = ra - rb
sub rz sp // rz -= sp
```

MUL - Multiplication

DIV - Division

OR - Logical OR

AND - Logical AND

XOR - Logical XOR

TEQ - Test if equal

TNE - Test if not equal

TLT - Test if strictly lower

TLE - Test if lower or equal

TGT - Test if strictly greater

TGE - Test if greater or equal

PUSH - Push register on stack

POP - Pop register from stack

STR - Store with direct addressing

LD - Load with direct addressing

STRP - Store with indirect addressing

LDP - Load with indirect addressing

XCHG - Exchange registers

ADDI - Addition with immediate

SUBI - Subtraction with immediate

ORI - Logical OR with immediate

ANDI - Logical AND with immediate

XORI - Logical XOR with immediate

TEQI - Test if equal with immediate

TNEI - Test if not equal with immediate

TLTI - Test if strictly lower with immediate

TLEI - Test if lower or equal with immediate

TGTI - Test if strictly greater with immediate

TGEI - Test if greater or equal with immediate

STRI - Store with direct immediate addressing

LDI - Load with direct immediate addressing

JZ - Relative jump if zero

JNZ - Relative jump if not zero

CALL - Call subroutine (Jump and link)

RET - Return from subroutine

JABS - Absolute jump