

Percival instruction set

Complete instruction set definition for RTL and Archc generator



Ronan Barzic

Allan Green

Percival instruction set
Complete instruction set definition for RTL and Archc generator
Edition 0

Author

Ronan Barzic

ronan.barzic@atmel.com

Author

Allan Green

allanalpha.green@gmail.com

Describe the Percival instruction set

1. Instruction set summary	1
1.1. Instruction formats	1
1.1.1. Type Lit	1
1.1.1.1. Format definition	1
1.1.1.2. Specification list	2
1.1.2. Type Jmp	2
1.1.2.1. Format definition	2
1.1.2.2. Specification list	3
1.1.3. Type Cond Jmp	3
1.1.3.1. Format definition	3
1.1.3.2. Specification list	4
1.1.4. Type Call	4
1.1.4.1. Format definition	4
1.1.4.2. Specification list	5
1.1.5. Type Alu	6
1.1.5.1. Format definition	6
1.1.5.2. Specification list	7
2. Detailed instruction set	9
2.1. add	9
2.1.1. Instruction definition	9
2.1.2. Instruction implementation	9
2.1.3. Assembler implementation	9
2.1.4. Specification list	10
2.2. and	10
2.2.1. Instruction definition	10
2.2.2. Instruction implementation	10
2.2.3. Assembler implementation	10
2.2.4. Specification list	10
2.3. call	11
2.3.1. Instruction definition	11
2.3.2. Instruction implementation	11
2.3.3. Assembler implementation	11
2.3.4. Specification list	11
2.4. cond_jmp	11
2.4.1. Instruction definition	11
2.4.2. Instruction implementation	12
2.4.3. Assembler implementation	12
2.4.4. Specification list	12
2.5. drop	12
2.5.1. Instruction definition	12
2.5.2. Instruction implementation	12
2.5.3. Assembler implementation	13
2.5.4. Specification list	13
2.6. dup	13
2.6.1. Instruction definition	13
2.6.2. Instruction implementation	13
2.6.3. Assembler implementation	14
2.6.4. Specification list	14
2.7. exit	14
2.7.1. Instruction definition	14
2.7.2. Instruction implementation	14
2.7.3. Assembler implementation	15
2.7.4. Specification list	15

2.8. invert	15
2.8.1. Instruction definition	15
2.8.2. Instruction implementation	15
2.8.3. Assembler implementation	16
2.8.4. Specification list	16
2.9. jmp	16
2.9.1. Instruction definition	16
2.9.2. Instruction implementation	16
2.9.3. Assembler implementation	16
2.9.4. Specification list	17
2.10. lit	17
2.10.1. Instruction definition	17
2.10.2. Instruction implementation	17
2.10.3. Assembler implementation	17
2.10.4. Specification list	17
2.11. mem_rd	18
2.11.1. Instruction definition	18
2.11.2. Instruction implementation	18
2.11.3. Assembler implementation	18
2.11.4. Specification list	18
2.12. mem_wr	19
2.12.1. Instruction definition	19
2.12.2. Instruction implementation	19
2.12.3. Assembler implementation	19
2.12.4. Specification list	19
2.13. n_eq_t	20
2.13.1. Instruction definition	20
2.13.2. Instruction implementation	20
2.13.3. Assembler implementation	20
2.13.4. Specification list	20
2.14. nip	21
2.14.1. Instruction definition	21
2.14.2. Instruction implementation	21
2.14.3. Assembler implementation	21
2.14.4. Specification list	21
2.15. n_lt_t	21
2.15.1. Instruction definition	21
2.15.2. Instruction implementation	22
2.15.3. Assembler implementation	22
2.15.4. Specification list	22
2.16. nop	22
2.16.1. Instruction definition	22
2.16.2. Instruction implementation	23
2.16.3. Assembler implementation	23
2.16.4. Specification list	23
2.17. n_sl_t	23
2.17.1. Instruction definition	23
2.17.2. Instruction implementation	23
2.17.3. Assembler implementation	24
2.17.4. Specification list	24
2.18. n_sr_t	24
2.18.1. Instruction definition	24
2.18.2. Instruction implementation	24
2.18.3. Assembler implementation	25

2.18.4. Specification list	25
2.19. n_ult_t	25
2.19.1. Instruction definition	25
2.19.2. Instruction implementation	25
2.19.3. Assembler implementation	26
2.19.4. Specification list	26
2.20. or	26
2.20.1. Instruction definition	26
2.20.2. Instruction implementation	26
2.20.3. Assembler implementation	27
2.20.4. Specification list	27
2.21. over	27
2.21.1. Instruction definition	27
2.21.2. Instruction implementation	27
2.21.3. Assembler implementation	28
2.21.4. Specification list	28
2.22. rs_cp	28
2.22.1. Instruction definition	28
2.22.2. Instruction implementation	28
2.22.3. Assembler implementation	29
2.22.4. Specification list	29
2.23. rs_pop	29
2.23.1. Instruction definition	29
2.23.2. Instruction implementation	29
2.23.3. Assembler implementation	30
2.23.4. Specification list	30
2.24. rs_push	30
2.24.1. Instruction definition	30
2.24.2. Instruction implementation	30
2.24.3. Assembler implementation	31
2.24.4. Specification list	31
2.25. stk_dep	31
2.25.1. Instruction definition	31
2.25.2. Instruction implementation	31
2.25.3. Assembler implementation	32
2.25.4. Specification list	32
2.26. sub1	32
2.26.1. Instruction definition	32
2.26.2. Instruction implementation	32
2.26.3. Assembler implementation	32
2.26.4. Specification list	33
2.27. swap	33
2.27.1. Instruction definition	33
2.27.2. Instruction implementation	33
2.27.3. Assembler implementation	33
2.27.4. Specification list	34
2.28. xor	34
2.28.1. Instruction definition	34
2.28.2. Instruction implementation	34
2.28.3. Assembler implementation	34
2.28.4. Specification list	34

Instruction set summary



Note to the reader

The following tables are autogenerated...

1.1. Instruction formats

1.1.1. Type Lit

1.1.1.1. Format definition

```
spec['stkpc']['inst_type']['Type_Lit']['format']= {
  'opcode1' : {'size' : 1,
               'offset' : 15,
               'decode' : True, # Field is used for decoding
              },
  'uk15' : {'size' : 15,
            'offset' : 0
            },
}

# default implementation
spec['stkpc']['inst_type']['Type_Lit']['impl']= {

  'pc' : {
    'next' : 'increment'
  }

  'alu' : {
    'sela' : 'st0',
    'selb' : 'zeros',
    'opcode' : 'or',
  },

  'st0' : {
    'next' : 'imm',
  },

  'data_stack' : {
    'we' : True,
  },

  'return_stack' : {
    'we' : False,
    'data_in' : 'pc'
  },

  'dsp' : {
    'next' : 'dsp_inc_1',
  },
}
```

```
    },
    'rsp' : {
        'next_rsp' : 'rsp',
    },
    'ram' : {
        'we' : False,
    },
}

spec['stkpc']['inst_type']['Type_Lit']['asm'] = '%exp';
spec['stkpc']['inst_type']['Type_Lit']['fields'] = 'uk15';
# just an helper for quick docbook generation - we could write some code to do it
# automatically...
spec['stkpc']['inst_type']['Type_Lit']['docbook'] = ('opcode1', 'uk15');
```

1.1.1.2. Specification list

```
<src:fragref linkend="src_inst_def_format_type_lit"></src:fragref>
```

1.1.2. Type Jump

1.1.2.1. Format definition

```
spec['stkpc']['inst_type']['Type_Jump']['format'] = {
    'opcode1' : {'size' : 1,
                 'offset' : 15,
                 'decode' : True, # Field is used for decoding
                },
    'opcode2' : {'size' : 2,
                 'offset' : 13,
                 'decode' : True, # Field is used for decoding
                },
    'uk13' : {'size' : 13,
              'offset' : 0
             },
}

# default implementation
spec['stkpc']['inst_type']['Type_Jump']['impl'] = {

    'pc' : {
        'next' : 'instr'
    }

    'alu' : {
        'sela' : 'st0',
        'selb' : 'zeros',
        'opcode' : 'or',
    },
}
```

```

'st0' : {
    'next' : 'alu_out',
},

'data_stack' : {
    'we' : True,
},

'return_stack' : {
    'we' : False,
    'data_in' : 'pc'
},

'dsp' : {
    'next' : 'dsp',
},

'rsp' : {
    'next_rsp' : 'rsp',
},
'ram' : {
    'we' : False,
},
}

spec['stkpc']['inst_type']['Type_Jump']['asm'] = '%exp';
spec['stkpc']['inst_type']['Type_Jump']['fields'] = 'uk13';
# just an helper for quick docbook generation - we could write some code to do it
# automatically...
spec['stkpc']['inst_type']['Type_Jump']['docbook'] = ('opcode1', 'opcode2', 'uk13');

```

1.1.2.2. Specification list

```
<src:fragref linkend="src_inst_def_format_type_jump"></src:fragref>
```

1.1.3. Type Cond Jump

1.1.3.1. Format definition

```

spec['stkpc']['inst_type']['Type_Cond_Jump']['format'] = {
    'opcode1' : { 'size' : 1,
                  'offset' : 15,
                  'decode' : True, # Field is used for decoding
                },
    'opcode2' : { 'size' : 2,
                  'offset' : 13,
                  'decode' : True, # Field is used for decoding
                },
    'uk13' : { 'size' : 13,
              'offset' : 0
            },
}

```

```
# default implementation
spec['stkpc']['inst_type']['Type_Cond_Jmp']['impl']= {

  'pc' : {
    'next' : 'cond_jump'
  }

  'alu' : {
    'sela' : 'st0',
    'selb' : 'zeros',
    'opcode' : 'or',
  },

  'st0' : {
    'next' : 'alu_out',
  },

  'data_stack' : {
    'we' : True,
  },

  'return_stack' : {
    'we' : False,
    'data_in' : 'pc'
  },

  'dsp' : {
    'next' : 'dsp_dec_1',
  },

  'rsp' : {
    'next_rsp' : 'rsp',
  },
  'ram' : {
    'we' : False,
  },
}

spec['stkpc']['inst_type']['Type_Cond_Jmp']['asm'] = '%exp';
spec['stkpc']['inst_type']['Type_Cond_Jmp']['fields']= 'uk13';
# just an helper for quick docbook generation - we could write some code to do it
# automatically...
spec['stkpc']['inst_type']['Type_Cond_Jmp']['docbook']= ('opcode1', 'opcode2', 'uk13');
```

1.1.3.2. Specification list

```
<src:fragref linkend="src_inst_def_format_type_cond_jump"></src:fragref>
```

1.1.4. Type Call

1.1.4.1. Format definition

```
spec['stkpc']['inst_type']['Type_Call']['format']= {
  'opcode1' : { 'size' : 1,
```

```

        'offset' : 15,
        'decode' : True, # Field is used for decoding
    },
    'opcode2' : {'size' : 2,
                'offset' : 13,
                'decode' : True, # Field is used for decoding
    },
    'uk13' : {'size' : 13,
             'offset' : 0
    },
}

# default implementation
spec['stkpc']['inst_type']['Type_Call']['impl']= {

    'pc' : {
        'next' : 'instr'
    }

    'alu' : {
        'sela' : 'st0',
        'selb' : 'zeros',
        'opcode' : 'or',
    },

    'st0' : {
        'next' : 'alu_out',
    },

    'data_stack' : {
        'we' : True,
    },

    'return_stack' : {
        'we' : True,
        'data_in' : 'pc'
    },

    'dsp' : {
        'next' : 'dsp',
    },

    'rsp' : {
        'next_rsp' : 'rsp_inc_1',
    },
    'ram' : {
        'we' : False,
    },
}

spec['stkpc']['inst_type']['Type_Call']['asm'] = '%exp';
spec['stkpc']['inst_type']['Type_Call']['fields']= 'uk13';
# just an helper for quick docbook generation - we could write some code to do it
automatically...
spec['stkpc']['inst_type']['Type_Call']['docbook']= ('opcode1','opcode2', 'uk13');

```

1.1.4.2. Specification list

```
<src:fragref linkend="src_inst_def_format_type_call"></src:fragref>
```

1.1.5. Type Alu

1.1.5.1. Format definition

```
spec['stkpc']['inst_type']['Type_Alu']['format']= {
  'opcode1' : {'size' : 1,
               'offset' : 15,
               'decode' : True, # Field is used for decoding
              },
  'opcode2' : {'size' : 2,
               'offset' : 13,
               'decode' : True, # Field is used for decoding
              },
  'opcode3' : {'size' : 13,
               'offset' : 0,
               'decode' : True, # Field is used for decoding
              },
}

# default implementation
spec['stkpc']['inst_type']['Type_Alu']['impl']= {
  'pc' : {
    'next' : 'increment'
  },
  'alu' : {
    'sela' : 'st1',
    'selb' : 'st0',
    'opcode' : 'or',
  },
  'st0' : {
    'next' : 'alu_out',
  },
  'data_stack' : {
    'we' : False,
  },
  'return_stack' : {
    'we' : False,
    'data_in' : 'pc'
  },
  'dsp' : {
    'next_dsp' : 'dsp',
  },
  'rsp' : {
    'next_rsp' : 'rsp',
  },
  'ram' : {
    'we' : False,
  },
}
```

```
}  
  
spec['stkpc']['inst_type']['Type_Alus']['asm'] = '%exp';  
spec['stkpc']['inst_type']['Type_Alus']['fields'] = 'uk13';  
# just an helper for quick docbook generation - we could write some code to do it  
# automatialy...  
spec['stkpc']['inst_type']['Type_Alus']['docbook'] = ('opcode1', 'opcode2', 'opcode3');
```

1.1.5.2. Specification list

```
<src:fragref linkend="src_inst_def_format_type_alu"></src:fragref>
```


Detailed instruction set



Note to the reader

The following sections contain description of all instructions used by the μ Sequencer. Behaviour of each instruction is described as a set of properties (expressed as a Python dictionary) that are used to generate Verilog code, Assembler and ArchC configuration code (for binutils tools like GAS...)

2.1. add

2.1.1. Instruction definition

```
spec['stkpc']['inst']['add'] = {
    'fullname' : 'add',
    'Description' : '16-bit bitwise add',
    'opcode1' : 0,
    'opcode2' : 3,
    'opcode3' : 515,
    'type' : 'Type_Alu',
}
```

2.1.2. Instruction implementation

```
spec['stkpc']['inst']['add']['impl'] = {
    'alu' : {
        'op' : 'add',
    },
    'dsp' : {
        'next_dsp' : 'dsp_dec_1',
    },
}
```

2.1.3. Assembler implementation

```
# ASM :
# No special implementation - following default for instruction type
```

2.1.4. Specification list

```
<src:fragref linkend="src_inst_def_add"></src:fragref>
<src:fragref linkend="src_inst_impl_add"></src:fragref>
<src:fragref linkend="src_inst_asm_add"></src:fragref>
```

2.2. and

2.2.1. Instruction definition

```
spec['stkpc']['inst']['and'] = {
  'fullname' : 'and',
  'Description' : '16-bit bitwise and',
  'opcode1' : 0,
  'opcode2' : 3,
  'opcode3' : 771,
  'type' : 'Type_Alu',
}
```

2.2.2. Instruction implementation

```
spec['stkpc']['inst']['and']['impl'] = {
  'alu' : {
    'op' : 'and',
  },
  'dsp' : {
    'next_dsp' : 'dsp_dec_1',
  },
}
```

2.2.3. Assembler implementation

```
# ASM :
# No special implementation - following default for instruction type
```

2.2.4. Specification list

```
<src:fragref linkend="src_inst_def_and"></src:fragref>
<src:fragref linkend="src_inst_impl_and"></src:fragref>
<src:fragref linkend="src_inst_asm_and"></src:fragref>
```

2.3. call

2.3.1. Instruction definition

```
spec['stkpc']['inst']['call'] = {
  'fullname' : 'call',
  'Description' : 'jumps to pc given by instruction and saves old pc in return stack',
  'opcode1' : 1,
  'opcode2' : 2,
  'type' : 'Type_Call',
}
```

2.3.2. Instruction implementation

```
spec['stkpc']['inst']['call']['impl'] = {
}
```

2.3.3. Assembler implementation

```
# ASM :
# No special implementation - following default for instruction type
```

2.3.4. Specification list

```
<src:fragref linkend="src_inst_def_call"></src:fragref>
<src:fragref linkend="src_inst_impl_call"></src:fragref>
<src:fragref linkend="src_inst_asm_call"></src:fragref>
```

2.4. cond_jump

2.4.1. Instruction definition

```
spec['stkpc']['inst']['cond_jump'] = {
  'fullname' : 'cond_jump',
  'Description' : 'jumps to new PC value given by instruction if top of stack equal zero',
  'opcode1' : 1,
```

```
'opcode2'    : 1,  
  'type'      : 'Type_Cond_Jmp',  
}
```

2.4.2. Instruction implementation

```
spec['stkpc']['inst']['cond_jump']['impl'] = {  
  
}
```

2.4.3. Assembler implementation

```
# ASM :  
# No special implementation - following default for instruction type
```

2.4.4. Specification list

```
<src:fragref linkend="src_inst_def_cond_jump"></src:fragref>  
<src:fragref linkend="src_inst_impl_cond_jump"></src:fragref>  
<src:fragref linkend="src_inst_asm_cond_jump"></src:fragref>
```

2.5. drop

2.5.1. Instruction definition

```
spec['stkpc']['inst']['drop'] = {  
  'fullname' : 'drop',  
  'Description' : 'drop top of data stack',  
  'opcode1'    : 0,  
  'opcode2'    : 3,  
  'opcode3'    : 259,  
  'type'       : 'Type_Alui',  
}
```

2.5.2. Instruction implementation

```
spec['stkpc']['inst']['drop']['impl'] = {
    'alu'      : {
        'sela' : 'st1',
        'selb' : 'zero',
        'opcode' : 'or',
    },
    'dsp' : {
        'next_dsp' : 'dsp_dec_1',
    },
}
```

2.5.3. Assembler implementation

```
# ASM :
# No special implementation - following default for instruction type
```

2.5.4. Specification list

```
<src:fragref linkend="src_inst_def_drop"></src:fragref>
<src:fragref linkend="src_inst_impl_drop"></src:fragref>
<src:fragref linkend="src_inst_asm_drop"></src:fragref>
```

2.6. dup

2.6.1. Instruction definition

```
spec['stkpc']['inst']['dup'] = {
    'fullname' : 'dup',
    'Description' : 'duplicate top of stack and push it to stack',
    'opcode1' : 0,
    'opcode2' : 3,
    'opcode3' : 129,
    'type' : 'Type_Alu',
}
```

2.6.2. Instruction implementation

```
spec['stkpc']['inst']['dup']['impl'] = {
    'alu'      : {
```

```
        'sela' : 'st0',
        'selb' : 'zero',
        'opcode' : 'or',
    },

'dsp' : {
    'next_dsp' : 'dsp_inc_1',
},

'data_stack' : {
    'we' : True,
},
}
```

2.6.3. Assembler implementation

```
# ASM :
# No special implementation - following default for instruction type
```

2.6.4. Specification list

```
<src:fragref linkend="src_inst_def_dup"></src:fragref>
<src:fragref linkend="src_inst_impl_dup"></src:fragref>
<src:fragref linkend="src_inst_asm_dup"></src:fragref>
```

2.7. exit

2.7.1. Instruction definition

```
spec['stkpc']['inst']['exit'] = {
    'fullname' : 'exit',
    'Description' : 'Jumps to instruction address given by top value of return stack',
    'opcode1' : 0,
    'opcode2' : 3,
    'opcode3' : 4108,
    'type' : 'Type_Alu',
}
```

2.7.2. Instruction implementation

```
spec['stkpc']['inst']['exit']['impl'] = {
    'alu' : {
```

```

    'sela' : 'st0',
    'selb' : 'zero',
    'opcode' : 'or',
  },
  'rsp' : {
    'next_rsp' : 'rsp_dec_1',
  },
}

```

2.7.3. Assembler implementation

```

# ASM :
# No special implementation - following default for instruction type

```

2.7.4. Specification list

```

<src:fragref linkend="src_inst_def_exit"></src:fragref>
<src:fragref linkend="src_inst_impl_exit"></src:fragref>
<src:fragref linkend="src_inst_asm_exit"></src:fragref>

```

2.8. invert

2.8.1. Instruction definition

```

spec['stkpc']['inst']['invert'] = {
  'fullname' : 'invert',
  'Description' : '16 bitwise invert',
  'opcode1' : 0,
  'opcode2' : 3,
  'opcode3' : 1539,
  'type' : 'Type_Alu',
}

```

2.8.2. Instruction implementation

```

spec['stkpc']['inst']['invert']['impl'] = {
  'alu' : {
    'sela' : 'st0',
    'selb' : 'ones',
    'opcode' : 'xor',
  },
}

```

```
}
```

2.8.3. Assembler implementation

```
# ASM :  
# No special implementation - following default for instruction type
```

2.8.4. Specification list

```
<src:fragref linkend="src_inst_def_invert"></src:fragref>  
<src:fragref linkend="src_inst_impl_invert"></src:fragref>  
<src:fragref linkend="src_inst_asm_invert"></src:fragref>
```

2.9. jmp

2.9.1. Instruction definition

```
spec['stkpc']['inst']['jmp'] = {  
  'fullname' : 'jmp',  
  'Description' : 'simple jump',  
  'opcode1' : 1,  
  'opcode2' : 0,  
  'type' : 'Type_Jmp',  
}
```

2.9.2. Instruction implementation

```
spec['stkpc']['inst']['jmp']['impl'] = {  
  
}
```

2.9.3. Assembler implementation

```
# ASM :  
# No special implementation - following default for instruction type
```

2.9.4. Specification list

```
<src:fragref linkend="src_inst_def_jump"></src:fragref>  
<src:fragref linkend="src_inst_impl_jump"></src:fragref>  
<src:fragref linkend="src_inst_asm_jump"></src:fragref>
```

2.10. lit

2.10.1. Instruction definition

```
spec['stkpc']['inst']['lit'] = {  
  'fullname' : 'lit',  
  'Description' : 'Load 15 bit literal to top of stack',  
  'opcode1' : 1,  
  'type' : 'Type_Lit',  
}
```

2.10.2. Instruction implementation

```
spec['stkpc']['inst']['lit']['impl'] = {  
  
}
```

2.10.3. Assembler implementation

```
# ASM :  
# No special implementation - following default for instruction type
```

2.10.4. Specification list

```
<src:fragref linkend="src_inst_def_lit"></src:fragref>  
<src:fragref linkend="src_inst_impl_lit"></src:fragref>
```

```
<src:fragref linkend="src_inst_asm_lit"></src:fragref>
```

2.11. mem_rd

2.11.1. Instruction definition

```
spec['stkpc']['inst']['mem_rd'] = {  
  'fullname' : 'mem_rd',  
  'Description' : 'read from memory',  
  'opcode1' : 0,  
  'opcode2' : 3,  
  'opcode3' : 3073,  
  'type' : 'Type_Alu',  
}
```

2.11.2. Instruction implementation

```
spec['stkpc']['inst']['mem_rd']['impl'] = {  
  
  'alu' : {  
    'sela' : 'st0',  
    'selb' : 'zero',  
    'opcode' : 'or',  
  },  
  
  'dsp' : {  
    'next_dsp' : 'dsp_inc_1',  
  },  
  
  'st0' : {  
    'next' : 'new_ram_io',  
  },  
  
  'data_stack' : {  
    'we' : True,  
  },  
}
```

2.11.3. Assembler implementation

```
# ASM :  
# No special implementation - following default for instruction type
```

2.11.4. Specification list

```
<src:fragref linkend="src_inst_def_mem_rd"></src:fragref>
<src:fragref linkend="src_inst_impl_mem_rd"></src:fragref>
<src:fragref linkend="src_inst_asm_mem_rd"></src:fragref>
```

2.12. mem_wr

2.12.1. Instruction definition

```
spec['stkpc']['inst']['mem_wr'] = {
  'fullname' : 'mem_wr',
  'Description' : 'memory write',
  'opcode1' : 0,
  'opcode2' : 3,
  'opcode3' : 291,
  'type' : 'Type_Alui',
}
```

2.12.2. Instruction implementation

```
spec['stkpc']['inst']['mem_wr']['impl'] = {
  'alu' : {
    'sela' : 'st0',
    'selb' : 'zero',
    'opcode' : 'or',
  },
  'dsp' : {
    'next_dsp' : 'dsp_dec_1',
  },
  'ram' : {
    'we' : True,
  },
}
```

2.12.3. Assembler implementation

```
# ASM :
# No special implementation - following default for instruction type
```

2.12.4. Specification list

```
<src:fragref linkend="src_inst_def_mem_wr"></src:fragref>
```

```
<src:fragref linkend="src_inst_impl_mem_wr"></src:fragref>
<src:fragref linkend="src_inst_asm_mem_wr"></src:fragref>
```

2.13. n_eq_t

2.13.1. Instruction definition

```
spec['stkpc']['inst']['n_eq_t'] = {
  'fullname' : 'n_eq_t',
  'Description' : 'Returns a 0 if next in stack not equal to top of stack, else returns
fff ',
  'opcode1'      : 0,
  'opcode2'      : 3,
  'opcode3'      : 1795,
  'type'         : 'Type_Alu',
}
```

2.13.2. Instruction implementation

```
spec['stkpc']['inst']['n_eq_t']['impl'] = {
  'alu' : {
    'sela' : 'st1',
    'selb' : 'st0',
    'opcode' : 'eq',
  },
  'dsp' : {
    'next_dsp' : 'dsp_dec_1',
  },
}
```

2.13.3. Assembler implementation

```
# ASM :
# No special implementation - following default for instruction type
```

2.13.4. Specification list

```
<src:fragref linkend="src_inst_def_n_eq_t"></src:fragref>
<src:fragref linkend="src_inst_impl_n_eq_t"></src:fragref>
<src:fragref linkend="src_inst_asm_n_eq_t"></src:fragref>
```

2.14. nip

2.14.1. Instruction definition

```
spec['stkpc']['inst']['nip'] = {
  'fullname' : 'nip',
  'Description' : 'no operation data stack pointers -1',
  'opcode1' : 0,
  'opcode2' : 3,
  'opcode3' : 3,
  'type' : 'Type_Alus',
}
```

2.14.2. Instruction implementation

```
spec['stkpc']['inst']['nip']['impl'] = {
  'alu' : {
    'sela' : 'st0',
    'selb' : 'zero',
    'opcode' : 'nip',
  },
}
```

2.14.3. Assembler implementation

```
# ASM :
# No special implementation - following default for instruction type
```

2.14.4. Specification list

```
<src:fragref linkend="src_inst_def_nip"></src:fragref>
<src:fragref linkend="src_inst_impl_nip"></src:fragref>
<src:fragref linkend="src_inst_asm_nip"></src:fragref>
```

2.15. n_lt_t

2.15.1. Instruction definition

```
spec['stkpc']['inst']['n_lt_t'] = {
```

```
'fullname' : 'n_lt_t',
'Description' : 'Return ffff if next of stack smaller than top of stack, else return 0',
'opcode1' : 0,
'opcode2' : 3,
'opcode3' : 2051,
'type' : 'Type_Alu',
}
```

2.15.2. Instruction implementation

```
spec['stkpc']['inst']['n_lt_t']['impl'] = {
  'alu' : {
    'sela' : 'st1',
    'selb' : 'st0',
    'opcode' : 'slt',
  },
  'dsp' : {
    'next_dsp' : 'dsp_dec_1',
  },
}
```

2.15.3. Assembler implementation

```
# ASM :
# No special implementation - following default for instruction type
```

2.15.4. Specification list

```
<src:fragref linkend="src_inst_def_n_lt_t"></src:fragref>
<src:fragref linkend="src_inst_impl_n_lt_t"></src:fragref>
<src:fragref linkend="src_inst_asm_n_lt_t"></src:fragref>
```

2.16. nop

2.16.1. Instruction definition

```
spec['stkpc']['inst']['nop'] = {
  'fullname' : 'nop',
  'Description' : 'no operation',
  'opcode1' : 0,
  'opcode2' : 3,
  'opcode3' : 0,
}
```

```
'type'      : 'Type_Alus',
}
```

2.16.2. Instruction implementation

```
spec['stkpc']['inst']['nop']['impl'] = {
  'alu'      : {
    'sela'   : 'st0',
    'selb'   : 'zero',
    'opcode' : 'nop',
  },
}
```

2.16.3. Assembler implementation

```
# ASM :
# No special implementation - following default for instruction type
```

2.16.4. Specification list

```
<src:fragref linkend="src_inst_def_nop"></src:fragref>
<src:fragref linkend="src_inst_impl_nop"></src:fragref>
<src:fragref linkend="src_inst_asm_nop"></src:fragref>
```

2.17. n_sl_t

2.17.1. Instruction definition

```
spec['stkpc']['inst']['n_sl_t'] = {
  'fullname' : 'n_sl_t',
  'Description' : 'next of stack is shifted left by value in top of stack',
  'opcode1'   : 0,
  'opcode2'   : 3,
  'opcode3'   : 3331,
  'type'      : 'Type_Alus',
}
```

2.17.2. Instruction implementation

```
spec['stkpc']['inst']['n_sl_t']['impl'] = {  
    'alu'      : {  
        'sela' : 'st1',  
        'selb' : 'st0',  
        'opcode' : 'sll',  
    },  
    'dsp' : {  
        'next_dsp' : 'dsp_dec_1',  
    },  
}
```

2.17.3. Assembler implementation

```
# ASM :  
# No special implementation - following default for instruction type
```

2.17.4. Specification list

```
<src:fragref linkend="src_inst_def_n_sl_t"></src:fragref>  
<src:fragref linkend="src_inst_impl_n_sl_t"></src:fragref>  
<src:fragref linkend="src_inst_asm_n_sl_t"></src:fragref>
```

2.18. n_sr_t

2.18.1. Instruction definition

```
spec['stkpc']['inst']['n_sr_t'] = {  
    'fullname' : 'n_sr_t',  
    'Description' : 'next of stack is shifted right by value in top of stack',  
    'opcode1' : 0,  
    'opcode2' : 3,  
    'opcode3' : 2307,  
    'type' : 'Type_Alu',  
}
```

2.18.2. Instruction implementation

```
spec['stkpc']['inst']['n_sr_t']['impl'] = {  
    'alu'      : {
```

```

    'sela' : 'st1',
    'selb' : 'st0',
    'opcode' : 'srl',
  },

  'dsp' : {
    'next_dsp' : 'dsp_dec_1',
  },
}

```

2.18.3. Assembler implementation

```

# ASM :
# No special implementation - following default for instruction type

```

2.18.4. Specification list

```

<src:fragref linkend="src_inst_def_n_sr_t"></src:fragref>
<src:fragref linkend="src_inst_impl_n_sr_t"></src:fragref>
<src:fragref linkend="src_inst_asm_n_sr_t"></src:fragref>

```

2.19. n_ult_t

2.19.1. Instruction definition

```

spec['stkpc']['inst']['n_ult_t'] = {
  'fullname' : 'n_ult_t',
  'Description' : 'n unisigned less than t',
  'opcode1' : 0,
  'opcode2' : 3,
  'opcode3' : 129,
  'type' : 'Type_Alu',
}

```

2.19.2. Instruction implementation

```

spec['stkpc']['inst']['n_ult_t']['impl'] = {
  'alu' : {
    'sela' : 'st1',
    'selb' : 'st0',
    'opcode' : 'sltu',

```

```
    },
    'dsp' : {
        'next_dsp' : 'dsp_dec_1',
    },
}
```

2.19.3. Assembler implementation

```
# ASM :
# No special implementation - following default for instruction type
```

2.19.4. Specification list

```
<src:fragref linkend="src_inst_def_n_ult_t"></src:fragref>
<src:fragref linkend="src_inst_impl_n_ult_t"></src:fragref>
<src:fragref linkend="src_inst_asm_n_ult_t"></src:fragref>
```

2.20. or

2.20.1. Instruction definition

```
spec['stkpc']['inst']['or'] = {
    'fullname' : 'or',
    'Description' : '16-bit bitwise or',
    'opcode1' : 0,
    'opcode2' : 3,
    'opcode3' : 1027,
    'type' : 'Type_Alui',
}
```

2.20.2. Instruction implementation

```
spec['stkpc']['inst']['or']['impl'] = {
    'alu' : {
        'op' : 'or',
    },
    'dsp' : {
        'next_dsp' : 'dsp_dec_1',
    },
}
```

2.20.3. Assembler implementation

```
# ASM :
# No special implementation - following default for instruction type
```

2.20.4. Specification list

```
<src:fragref linkend="src_inst_def_or"></src:fragref>
<src:fragref linkend="src_inst_impl_or"></src:fragref>
<src:fragref linkend="src_inst_asm_or"></src:fragref>
```

2.21. over

2.21.1. Instruction definition

```
spec['stkpc']['inst']['over'] = {
  'fullname' : 'over',
  'Description' : 'writes next the value of next in stack to the new top of stack',
  'opcode1' : 0,
  'opcode2' : 3,
  'opcode3' : 385,
  'type' : 'Type_Al',
}
```

2.21.2. Instruction implementation

```
spec['stkpc']['inst']['over']['impl'] = {
  'alu' : {
    'sela' : 'st1',
    'selb' : 'zero',
    'opcode' : 'or',
  },
  'dsp' : {
    'next_dsp' : 'dsp_inc_1',
  },
  'data_stack' : {
    'we' : True,
  },
}
```

2.21.3. Assembler implementation

```
# ASM :  
# No special implementation - following default for instruction type
```

2.21.4. Specification list

```
<src:fragref linkend="src_inst_def_over"></src:fragref>  
<src:fragref linkend="src_inst_impl_over"></src:fragref>  
<src:fragref linkend="src_inst_asm_over"></src:fragref>
```

2.22. rs_cp

2.22.1. Instruction definition

```
spec['stkpc']['inst']['rs_cp'] = {  
    'fullname' : 'rs_cp',  
    'Description' : 'copy top or return stack to top of data stack',  
    'opcode1' : 0,  
    'opcode2' : 3,  
    'opcode3' : 2945,  
    'type' : 'Type_Alu',  
}
```

2.22.2. Instruction implementation

```
spec['stkpc']['inst']['rs_cp']['impl'] = {  
    'alu' : {  
        'sela' : 'rst0',  
        'selb' : 'zero',  
        'opcode' : 'or',  
    },  
    'dsp' : {  
        'next_dsp' : 'dsp_inc_1',  
    },  
    'data_stack' : {  
        'we' : True,  
    },  
}
```

2.22.3. Assembler implementation

```
# ASM :
# No special implementation - following default for instruction type
```

2.22.4. Specification list

```
<src:fragref linkend="src_inst_def_rs_cp"></src:fragref>
<src:fragref linkend="src_inst_impl_rs_cp"></src:fragref>
<src:fragref linkend="src_inst_asm_rs_cp"></src:fragref>
```

2.23. rs_pop

2.23.1. Instruction definition

```
spec['stkpc']['inst']['rs_pop'] = {
  'fullname' : 'rs_pop',
  'Description' : 'Pop value from return stack',
  'opcode1' : 0,
  'opcode2' : 3,
  'opcode3' : 2957,
  'type' : 'Type_Alu',
}
```

2.23.2. Instruction implementation

```
spec['stkpc']['inst']['rs_pop']['impl'] = {
  'alu' : {
    'sela' : 'rst0',
    'selb' : 'zero',
    'opcode' : 'or',
  },
  'dsp' : {
    'next_dsp' : 'dsp_inc_1',
  },
  'rsp' : {
    'next_rsp' : 'rsp_dec_1',
  },
  'return_stack' : {
    'data_in' : 'st0',
  },
  'data_stack' : {
    'we' : True,
  },
}
```

```
    },  
  }
```

2.23.3. Assembler implementation

```
# ASM :  
# No special implementation - following default for instruction type
```

2.23.4. Specification list

```
<src:fragref linkend="src_inst_def_rs_pop"></src:fragref>  
<src:fragref linkend="src_inst_impl_rs_pop"></src:fragref>  
<src:fragref linkend="src_inst_asm_rs_pop"></src:fragref>
```

2.24. rs_push

2.24.1. Instruction definition

```
spec['stkpc']['inst']['rs_push'] = {  
  'fullname' : 'rs_push',  
  'Description' : 'push to register stack',  
  'opcode1' : 0,  
  'opcode2' : 3,  
  'opcode3' : 327,  
  'type' : 'Type_Alu',  
}
```

2.24.2. Instruction implementation

```
spec['stkpc']['inst']['rs_push']['impl'] = {  
  'alu' : {  
    'sela' : 'st1',  
    'selb' : 'zero',  
    'opcode' : 'or',  
  },  
  'dsp' : {  
    'next_dsp' : 'dsp_dec_1',  
  },  
  'rsp' : {  
    'next_rsp' : 'rsp_inc_1',  
  },  
}
```

```

    },
    'return_stack' : {
        'we' : True,
        'data_in' : 'st0'
    },
}

```

2.24.3. Assembler implementation

```

# ASM :
# No special implementation - following default for instruction type

```

2.24.4. Specification list

```

<src:fragref linkend="src_inst_def_rs_push"></src:fragref>
<src:fragref linkend="src_inst_impl_rs_push"></src:fragref>
<src:fragref linkend="src_inst_asm_rs_push"></src:fragref>

```

2.25. stk_dep

2.25.1. Instruction definition

```

spec['stkpc']['inst']['stk_dep'] = {
    'fullname' : 'stk_dep',
    'Description' : 'Put stacks depths concatenated in the top of stack (rsp+dsp)',
    'opcode1' : 0,
    'opcode2' : 3,
    'opcode3' : 3584,
    'type' : 'Type_Alus',
}

```

2.25.2. Instruction implementation

```

spec['stkpc']['inst']['stk_dep']['impl'] = {
    'alu' : {
        'sela' : 'rsp',
        'selb' : 'dsp',
        'opcode' : 'add',
    },
}

```

2.25.3. Assembler implementation

```
# ASM :  
# No special implementation - following default for instruction type
```

2.25.4. Specification list

```
<src:fragref linkend="src_inst_def_stk_dep"></src:fragref>  
<src:fragref linkend="src_inst_impl_stk_dep"></src:fragref>  
<src:fragref linkend="src_inst_asm_stk_dep"></src:fragref>
```

2.26. sub1

2.26.1. Instruction definition

```
spec['stkpc']['inst']['sub1'] = {  
  'fullname' : 'sub1',  
  'Description' : 'Subtracts 1 from top of stack',  
  'opcode1' : 0,  
  'opcode2' : 3,  
  'opcode3' : 2563,  
  'type' : 'Type_Alu',  
}
```

2.26.2. Instruction implementation

```
spec['stkpc']['inst']['sub1']['impl'] = {  
  'alu' : {  
    'sela' : 'st0',  
    'selb' : 'ones',  
    'opcode' : 'sub_1',  
  },  
}
```

2.26.3. Assembler implementation

```
# ASM :  
# No special implementation - following default for instruction type
```

2.26.4. Specification list

```
<src:fragref linkend="src_inst_def_sub1"></src:fragref>  
<src:fragref linkend="src_inst_impl_sub1"></src:fragref>  
<src:fragref linkend="src_inst_asm_sub1"></src:fragref>
```

2.27. swap

2.27.1. Instruction definition

```
spec['stkpc']['inst']['swap'] = {  
  'fullname' : 'swap',  
  'Description' : 'swap top and next of stack',  
  'opcode1' : 0,  
  'opcode2' : 3,  
  'opcode3' : 384,  
  'type' : 'Type_Alu',  
}
```

2.27.2. Instruction implementation

```
spec['stkpc']['inst']['swap']['impl'] = {  
  'alu' : {  
    'sela' : 'st1',  
    'selb' : 'zero',  
    'opcode' : 'or',  
  },  
  'data_stack' : {  
    'we' : True,  
  },  
}
```

2.27.3. Assembler implementation

```
# ASM :  
# No special implementation - following default for instruction type
```

2.27.4. Specification list

```
<src:fragref linkend="src_inst_def_swap"></src:fragref>  
<src:fragref linkend="src_inst_impl_swap"></src:fragref>  
<src:fragref linkend="src_inst_asm_swap"></src:fragref>
```

2.28. xor

2.28.1. Instruction definition

```
spec['stkpc']['inst']['xor'] = {  
    'fullname' : 'xor',  
    'Description' : '16-bit bitwise xor',  
    'opcode1' : 0,  
    'opcode2' : 3,  
    'opcode3' : 1283,  
    'type' : 'Type_Alu',  
}
```

2.28.2. Instruction implementation

```
spec['stkpc']['inst']['xor']['impl'] = {  
    'alu' : {  
        'op' : 'xor',  
    },  
    'dsp' : {  
        'next_dsp' : 'dsp_dec_1',  
    },  
}
```

2.28.3. Assembler implementation

```
# ASM :  
# No special implementation - following default for instruction type
```

2.28.4. Specification list

```
<src:fragref linkend="src_inst_def_xor"></src:fragref>  
<src:fragref linkend="src_inst_impl_xor"></src:fragref>  
<src:fragref linkend="src_inst_asm_xor"></src:fragref>
```

Index

