CTM16 instruction set

Complete instruction set definition for RTL and Archc generator

Ronan Barzic

CTM16 instruction set
Complete instruction set definition for RTL and Archc generator
Edition 0

Author Ronan Barzic ronan.barzic@atmel.com.com

Describe the CTM16 instruction set

Preface

Instruction set summary

Note to the reader

The following tables are autogenerated...

1.1. Instruction table

Table 1.1. Instruction summary

Name	Function	Type	[15..12]	[11..8]	[7..4]	[3..0]
add2	16-bit signed addition	I	0x1	rd	rs1	uk4
add2c	16-bit addition with carry	III	0x7	rd	0x8	rs2
add3	16-bit signed addition	VII	0xc	rd	rs1	rs2
and	16-bit bitwise and	III	0x7	rd	0x0	rs2
andr1	Bitwise and with r1 and immediate value	IV	0x8	rd	0x0	uk4
br	Branch if condition, address from a register	VI	0xa	cond	rd	k4
brpc	Branch if condition, relative to pc	V	0x9	cond	k8[7..0]	k8[3..0]
halt	Halt the CPU	VIII	0xe	rd	rs1	0x3
lb	Load byte	II	0x4	rd	rs1	uk4
ldir1	Load 12-bit immediate value in upper part of r1	IX	0xf	k12[11..8]	k12[7..4]	k12[3..0]
lw	Load word	II	0x3	rd	rs1	uk4
mfsr	Move from special register	VIII	0xe	rd	rs1	0x1
mtsr	Move to special register	VIII	0xe	rd	rs1	0x0
or	16-bit bitwise or	III	0x7	rd	0x1	rs2
ori4	immediate OR with 4-bit value	I	0x0	rd	rs1	uk4
orr1	Bitwise or with r1 and immediate value	IV	0x8	rd	0x1	uk4
ret	Return from function call	VIII	0xe	rd	rs1	0x2
sb	Store byte	II	0x6	rd	rs1	uk4
sext	8-bit to 16-bit Sign extention	III	0x7	rd	0x7	rs2
sll	16-bit Shift Left, Logical	III	0x7	rd	0x6	rs2
slli	Shift Left, Logical, using immediate value	IV	0x8	rd	0x7	uk4
sra	16-bit Shift Right, arithmetic	III	0x7	rd	0x4	rs2
srai	Shift Right, Arithmetic, using immediate value	IV	0x8	rd	0x5	uk4
srl	16-bit Shift Right, Logical	III	0x7	rd	0x5	rs2

Name	Function	Type	[15..12]	[11..8]	[7..4]	[3..0]
srli	Shift Right, Logical, using immediate value	IV	0x8	rd	0x6	uk4
sub2	16-bit signed substraction	I	0x2	rd	rs1	uk4
sub2c	16-bit substraction with carry	III	0x7	rd	0x9	rs2
sub3	16-bit signed addition	VII	0xd	rd	rs1	rs2
sw	Store word	II	0x5	rd	rs1	uk4
xnor	16-bit bitwise negated exclusive or	III	0x7	rd	0x3	rs2
xnorr1	Bitwise xnor with r1 and immediate value	IV	0x8	rd	0x3	uk4
xor	16-bit bitwise exclusive or	III	0x7	rd	0x2	rs2
xorr1	Bitwise xor with r1 and immediate value	IV	0x8	rd	0x2	uk4

1.2. Condition code table

Table 1.2. Condition code

Name	Value	Function
al	14	Always
alal	15	Always and link
c	10	Carry set
eq	0	Equal (signed)
ge	5	Greater or equal (signed)
geu	9	Greater or equal (unsigned)
gt	4	Greater than (signed)
gtu	8	Greater than (unsigned)
le	3	Less or equal (signed)
leu	7	Less or equal (unsigned)
lt	2	Less than (signed)
ltu	6	Less than (unsigned)
neq	1	Not equal (signed)
o	11	Overflow (signed)
ou	0	Overflow (unsigned)

1.3. Instruction formats

1.3.1. Type I

1.3.1.1. Format definition

```
spec['useq']['inst_type']['Type_I']['format']= {
    'opcode1' : {'size'   : 4,
                'offset' : 12,
                 'decode' : True,  # Field is used for decoding
                },
    'rd'     : {'size'   : 4,
                'offset' : 8
                },
    'rs1'     : {'size'   : 4,
                'offset'  : 4
                },
    'uk4'     : {'size'   : 4,
                'offset'  : 0
                },
}




# default implementation
spec['useq']['inst_type']['Type_I']['impl']= {
    'rf' : {
        'sela' : 'rd',
        'selb' : 'rs1',
        'selc' : 'rd',
    },

    'writeback'   : {
        'source' : "alu",
        'dest'   : "rfc[rd]",
    },
    'pc' : {
        'next' : 'increment'
    }
}

spec['useq']['inst_type']['Type_I']['asm']   =  '%reg, %reg, %exp';
spec['useq']['inst_type']['Type_I']['fields']=  'rd, rs1, uk4';
# just an helper for quick docbook generation - we could write some code to do it
 automatically...
spec['useq']['inst_type']['Type_I']['docbook']=  ('opcode1','rd', 'rs1', 'uk4');
```

1.3.1.2. Specification list

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

1.3.2. Type II

1.3.2.1. Format definition

```
spec['useq']['inst_type']['Type_II']['format']= {
    'opcode1' : {'size'   : 4,
                 'offset' : 12,
                 'decode' : True,  # Field is used for decoding
             },
    'rd'     : {'size'   : 4,
                'offset' : 8
            },
    'rs1'     : {'size'   : 4,
                 'offset'  : 4
                },
    'k4'     : {'size'   : 4,
                'offset'  : 0,
                'signed'  : True
                },
}

spec['useq']['inst_type']['Type_II']['impl']= {
    'pc' : {
        'next' : 'increment'
    },
    'flag'        : {
        'carry_in' : 0,
        'carry'    : 'keep',
    },

    'rf' : {
        'sela' : 'rs1',
        'selb' : 'rs1',  # default..?
        'selc' : 'rd',   # default..?
    },

}

spec['useq']['inst_type']['Type_II']['asm']   =  '%reg, %reg, %exp';
spec['useq']['inst_type']['Type_II']['fields']=  'rd, rs1, k4';

spec['useq']['inst_type']['Type_II']['docbook']=  ('opcode1','rd', 'rs1', 'uk4');
```

1.3.2.2. Specification list

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

1.3.3. Type III

1.3.3.1. Format definition

```
spec['useq']['inst_type']['Type_III']['format']= {
    'opcode1' : {'size'   : 4,
                 'offset' : 12,
                 'decode' : True,  # Field is used for decoding
             },
    'rd'     : {'size'   : 4,
                'offset' : 8
            },
    'opcode3'  : {'size'   : 4,
                 'offset'  : 4,
                  'decode' : True,  # Field is used for decoding
                },
    'rs2'     : {'size'   : 4,
                'offset'  : 0,
                },


}

spec['useq']['inst_type']['Type_III']['impl']= {
    'pc' : {
        'next' : 'increment'
    },
    'rf' : {
        'sela' : 'rd',
        'selb' : 'rs2',
        'selc' : 'rd',
    },
    'writeback'   : {
        'source' : "alu",
        'dest'   : "rfc[rd]",
    },
    'flag'        : { # some type_III instructions will override that
        'carry_in' : 0,
        'carry'    : 'keep',
    },
}

spec['useq']['inst_type']['Type_III']['asm']   =  '%reg, %reg';
spec['useq']['inst_type']['Type_III']['fields']=  'rd, rs2';

spec['useq']['inst_type']['Type_III']['docbook']=  ('opcode1','rd', 'opcode3', 'rs2');
```

1.3.3.2. Specification list

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

1.3.4. Type IV

1.3.4.1. Format definition

```
spec['useq']['inst_type']['Type_IV']['format']= {
    'opcode1' : {'size'   : 4,
                 'offset' : 12,
                 'decode' : True,  # Field is used for decoding
             },
    'rd'     : {'size'   : 4,
                'offset' : 8
            },
    'opcode3'  : {'size'   : 4,
                  'offset'  : 4,
                   'decode' : True,  # Field is used for decoding
                },
    'uk4'     : {'size'   : 4,
                'offset'  : 0,
                },
}

spec['useq']['inst_type']['Type_IV']['impl']= {
    'rf' : {
        'sela' : 'rd',
        'selb' : 'r1', # yes , we force r1 usage here
        'selc' : 'rd',
    },
    'pc' : {
        'next' : 'increment'
    },
    'writeback'   : {
        'source' : "alu",
        'dest'   : "rfc[rd]",
    },
    'flag'        : { # some type_IV instructions will override that
        'carry_in' : 0,
        'carry'    : 'keep',
    },

}

spec['useq']['inst_type']['Type_IV']['asm']   =  '%reg, %exp';

spec['useq']['inst_type']['Type_IV']['fields']=  'rd, uk4';

spec['useq']['inst_type']['Type_IV']['docbook']=  ('opcode1','rd', 'opcode3', 'uk4');
```

1.3.4.2. Specification list

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

1.3.5. Type V

1.3.5.1. Format definition

```
spec['useq']['inst_type']['Type_V']['format']= {
    'opcode1' : {'size'   : 4,
                 'offset' : 12,
                 'decode' : True,  # Field is used for decoding
                },
    'cond'    : {'size'   : 4,
                 'offset' : 8,
                 'decode' : True,  # Field is used for decoding
                },
    'k8'     : {'size'   : 8,
                'offset' : 0,
                'signed'  : True
                },
}

spec['useq']['inst_type']['Type_V']['impl']= {
    'rf' : {
        'sela' : 'rd',
        'selb' : 'rs1', # default - we don't use the RF
        'selc' : 'rs2',
    },
    'writeback'   : {
        'source' : "alu",
        'dest'   : "pc",
    },
    'flag'        : { # some type_IV instructions will override that
        'carry_in' : 0,
        'carry'    : 'keep',
    },

}


spec['useq']['inst_type']['Type_V']['asm']   =  '%exp';
spec['useq']['inst_type']['Type_V']['fields']=  'k8';

spec['useq']['inst_type']['Type_V']['docbook']=  ('opcode1','cond', 'k8[7..0]','k8[3..0]');
```

1.3.5.2. Specification list

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

1.3.6. Type VI

1.3.6.1. Format definition

```
spec['useq']['inst_type']['Type_VI']['format']= {
    'opcode1' : {'size'   : 4,
                 'offset' : 12,
                 'decode' : True,  # Field is used for decoding
                },
    'cond'    : {'size'   : 4,
                 'offset' : 8,
                 'decode' : True,  # Field is used for decoding
                },

    'rd'     : {'size'   : 4,
               'offset' : 4,
            },
    'uk4'    : {'size'   : 4,
                'offset' : 0
             },
}

spec['useq']['inst_type']['Type_VI']['impl']= {
    'rf' : {
        'sela' : 'r0',
        'selb' : 'rs1',
        'selc' : 'r0',
    },
    'writeback'   : {
        'source' : "alu",
        'dest'   : "pc",
    },
    'flag'        : { # some type_IV instructions will override that
        'carry_in' : 0,
        'carry'    : 'keep',
    },
  }


spec['useq']['inst_type']['Type_VI']['asm']    =  '%reg,%exp';
spec['useq']['inst_type']['Type_VI']['fields']=  'rd,uk4';

spec['useq']['inst_type']['Type_VI']['docbook']=  ('opcode1','cond', 'rd','k4');
```

1.3.6.2. Specification list

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

1.3.7. Type VII

1.3.7.1. Format definition

```
spec['useq']['inst_type']['Type_VII']['format']= {

    'opcode1' : {'size'   : 4,
                'offset' : 12,
                'decode' : True,  # Field is used for decoding
                },
    'rd'     : {'size'   : 4,
                'offset' : 8
                },
    'rs1' : {'size'   : 4,
             'offset' : 4,
                },
    'rs2'     : {'size'   : 4,
                'offset'  : 0
                },
}


spec['useq']['inst_type']['Type_VII']['impl']= {
    'rf' : {
        'sela' : 'rs1',
        'selb' : 'rs2',
        'selc' : 'rd',
        },
    'pc' : {
        'next' : 'increment',
        },
     'rf' : {
        'sela' : 'rs1',
        'selb' : 'rs2',
        'selc' : 'rd',
    },
    'writeback'   : {
        'source' : "alu",
        'dest'   : "rfc[rd]",
        },

    }
```

```
spec['useq']['inst_type']['Type_VII']['asm']    =  '%reg,%reg,%reg';
spec['useq']['inst_type']['Type_VII']['fields']=  'rd,rs1,rs2';
spec['useq']['inst_type']['Type_VII']['docbook']=  ('opcode1','rd', 'rs1','rs2');
```

1.3.7.2. Specification list

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

1.3.8. Type VIII

1.3.8.1. Format definition

```
spec['useq']['inst_type']['Type_VIII']['format']= {

    'opcode1' : {'size'   : 4,
                 'offset' : 12,
                 'decode' : True,  # Field is used for decoding
                 },
    'rd'     : {'size'   : 4,
                 'offset' : 8
                 },
    'rs1' : {'size'   : 4,
              'offset' : 4,
                 },
    'opcode4' : {'size'   : 4,
                 'offset'  : 0,
                 'decode' : True,  # Field is used for decoding
                 },
}


spec['useq']['inst_type']['Type_VIII']['impl']= {
    'pc' : {
        'next' : 'increment',
        },

    }

# custom asm per instruction is probably needed
spec['useq']['inst_type']['Type_VIII']['asm']    =  '%reg,%reg';
spec['useq']['inst_type']['Type_VIII']['fields']=  'rd,rs1';

spec['useq']['inst_type']['Type_VIII']['docbook']=  ('opcode1','rd', 'rs1','opcode4');
```

1.3.8.2. Specification list

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

1.3.9. Type IX

1.3.9.1. Format definition

```
# for the ldir1 instruction
spec['useq']['inst_type']['Type_IX']['format']= {

    'opcode1' : {'size'   : 4,
                 'offset' : 12,
                 'decode' : True,  # Field is used for decoding
                 },
    'k12'     : {'size'   : 12,
                 'offset'  : 0,
                 },
}


spec['useq']['inst_type']['Type_IX']['impl']= {
    'pc' : {
        'next' : 'increment',
        },

    }

# custom asm per instruction is probably needed
spec['useq']['inst_type']['Type_IX']['asm']   =  '%exp';
spec['useq']['inst_type']['Type_IX']['fields']=  'k12';

spec['useq']['inst_type']['Type_IX']['docbook']=  ('opcode1','k12[11..8]',
 'k12[7..4]','k12[3..0]');
```

1.3.9.2. Specification list

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

Detailled instruction set

Note to the reader

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

2.1. add2c

2.1.1. Instruction definition

```
spec['useq']['inst']['add2c'] =  {
    'fullname' : 'add2c',
    'Description' : '16-bit addition with carry ',
    'opcode1'      : 7,
    'opcode3'      : 8,
    'type'        : 'Type_III',

}
```

2.1.2. Instruction implementation

```
spec['useq']['inst']['add2c']['impl'] =  {

    'alu'         : {
        'op' : 'add',
        'sourcea' : 'rfa[rd]',
        'sourceb' : 'rfb[rs2]',
        },

    'flag'        : {
        'carry_in' : 'carry',
        'carry'    : 'update',
    }
}
```

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_add2c"></src:fragref>
<src:fragref linkend="src_inst_impl_add2c"></src:fragref>
<src:fragref linkend="src_inst_asm_add2c"></src:fragref>
```

2.2. add2

2.2.1. Instruction definition

```
spec['useq']['inst']['add2'] =  {
    'fullname' : 'add2',
    'Description' : '16-bit signed addition',
    'opcode1'      : 1,
    'type'        : 'Type_I',

}
```

2.2.2. Instruction implementation

```
spec['useq']['inst']['add2']['impl'] =  {

    'alu'         : {
        'op' : 'add',
        'sourcea' : 'rfa[rd]',
        'sourceb' : 'rfb[rs1]_or_uk4',
        },

    'flag'        : {
        'carry_in' : 0,
        'carry'    : 'update',
    }
}

#spec['useq']['inst']['add']['asm'] =  {
#    'main'         : "add(rf[rs1],rf[rs2])",
#    'variant1'     : "add(rf[rs1],rf[rs2])",
#}
```

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_add2"></src:fragref>
<src:fragref linkend="src_inst_impl_add2"></src:fragref>
<src:fragref linkend="src_inst_asm_add2"></src:fragref>
```

2.3. add3

2.3.1. Instruction definition

```
spec['useq']['inst']['add3'] =  {
    'fullname' : 'add3',
    'Description' : '16-bit signed addition',
    'opcode1'      : 0xC,
    'type'        : 'Type_VII',

}
```

2.3.2. Instruction implementation

```
spec['useq']['inst']['add3']['impl'] =  {

    'alu'         : {
        'op' : 'add',
        'sourcea' : 'rfa[rs1]',
        'sourceb' : 'rfb[rs2]',
        },
    'flag'        : { # some type_IV instructions will override that
        'carry_in' : 0,
        'carry'    : 'update',
    },

}
```

2.3.3. Assembler implementation

```
# Special variant to support expression instead of "pure" register name
# - To be used in gas macro
spec['useq']['inst']['add3']['asm'] =  {
    'main'         : '\"add3 %reg,%reg,%reg\"' +' ,rd,rs1,rs2',
    'variant1'     : '\"add3 %exp,%exp,%exp\"' + ',rd,rs1,rs2',
}
```

2.3.4. Specification list

```
<src:fragref linkend="src_inst_def_add3"></src:fragref>
<src:fragref linkend="src_inst_impl_add3"></src:fragref>
<src:fragref linkend="src_inst_asm_add3"></src:fragref>
```

2.4. andr1

2.4.1. Instruction definition

```
spec['useq']['inst']['andr1'] =  {
    'fullname' : 'andr1',
    'Description' : 'Bitwise and with r1 and  immediate value',
    'opcode1'      : 0x8,
    'opcode3'      : 0x0,
    'type'        : 'Type_IV',

}
```

2.4.2. Instruction implementation

```
spec['useq']['inst']['andr1']['impl'] =  {
    'alu'         : {
        'op' : 'and',
        'sourcea' : 'rfa[rd]',
        'sourceb' : 'rfb[r1]_or_uk4',
        },
}
```

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_andr1"></src:fragref>
<src:fragref linkend="src_inst_impl_andr1"></src:fragref>
<src:fragref linkend="src_inst_asm_andr1"></src:fragref>
```

2.5. and

2.5.1. Instruction definition

```
spec['useq']['inst']['and'] =  {
    'fullname' : 'and',
    'Description' : '16-bit bitwise and',
    'opcode1'      : 7,
    'opcode3'      : 0,
    'type'        : 'Type_III',

}
```

2.5.2. Instruction implementation

```
spec['useq']['inst']['and']['impl'] =  {

    'alu'         : {
        'op' : 'and',
        'sourcea' : 'rfa[rd]',
        'sourceb' : 'rfb[rs2]',
        },

    'flag'        : {
        'carry_in' : 0,
        'carry'    : 'keep',
    }
}
```

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

2.6. brpc

2.6.1. Instruction definition

```
spec['useq']['inst']['brpc'] =  {
    'fullname' : 'brpc',
    'Description' : 'Branch if condition, relative to pc',
    'opcode1'      : 0x9,
    'type'        : 'Type_V',

}
```

2.6.2. Instruction implementation

```
spec['useq']['inst']['brpc']['impl'] =  {
    'alu'         : {
        'op' : 'add',
        'sourcea' : 'npc',
        'sourceb' : 'k8',
        },
    'pc' : {
        'next' : 'cond_load_from_alu'
    },
    'writeback'   : {
        'source' : "alu",
        'dest'   : "cond_link_reg", # if condition is F, then this is an unconditionnal
 branch, with link
    },
}
```

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_brpc"></src:fragref>
<src:fragref linkend="src_inst_impl_brpc"></src:fragref>
```

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

2.7. br

2.7.1. Instruction definition

```
spec['useq']['inst']['br'] =  {
    'fullname' : 'br',
    'Description' : 'Branch if condition, address from a register',
    'opcode1'      : 0xA,
    'type'        : 'Type_VI',

}
```

2.7.2. Instruction implementation

```
spec['useq']['inst']['br']['impl'] =  {
    'alu'         : {
        'op' : 'add',
        'sourcea' : 'rfa[r0]',  # set in the instruction format definition
        'sourceb' : 'rfb[rs1]_or_uk4',
        },
    'pc' : {
        'next' : 'cond_load_from_alu'
    },
    'writeback'   : {
        'source' : "alu",
        'dest'   : "cond_link_reg", # if condition is F, then this is an unconditionnal
 branch, with link
    },
}
```

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_br"></src:fragref>
<src:fragref linkend="src_inst_impl_br"></src:fragref>
<src:fragref linkend="src_inst_asm_br"></src:fragref>
```

2.8. halt

2.8.1. Instruction definition

```
spec['useq']['inst']['halt'] =  {
    'fullname' : 'halt',
    'Description' : 'Halt the CPU ',
    'opcode1'      : 0xE,
    'opcode4'      : 0x3,
    'type'        : 'Type_VIII',

}
```

2.8.2. Instruction implementation

```
spec['useq']['inst']['halt']['impl'] =  {
    'rf' : {
        'sela' : 'r0', #  Don't care
        'selb' : 'rs1', # Don't care
        'selc' : 'r0', #  Don't care
        },

    'alu'         : {
        'op' : 'add',
        'sourcea' : 'rfa[r0]', # should be 0
        'sourceb' : 'rfb[rs1]',
        },
    'writeback'   : {
        'source' : 'alu',
        'dest'   : 'rfc', # r0
    },
    'flag'        : {
        'carry_in' : 0,
        'carry'    : 'keep',
    },
    'pc' : {
        'next' : 'increment'
    },
    'flow' : {
        'halt' : 'yes'  # we stop...
        },
}
```

2.8.3. Assembler implementation

```
spec['useq']['inst']['halt']['asm'] =  {
    'main'          : '\"halt %exp\"' +' ,rd',
}
```

2.8.4. Specification list

```
<src:fragref linkend="src_inst_def_halt"></src:fragref>
<src:fragref linkend="src_inst_impl_halt"></src:fragref>
<src:fragref linkend="src_inst_asm_halt"></src:fragref>
```

2.9. lb

2.9.1. Instruction definition

```
spec['useq']['inst']['lb'] =  {
    'fullname' : 'lb',
    'Description' : 'Load byte',
    'opcode1'      : 4,
    'type'        : 'Type_II',

}
```

2.9.2. Instruction implementation

```
spec['useq']['inst']['lb']['impl'] =  {

    'alu'         : {
        'op' : 'add',
        'sourcea' : 'rfa[rs1]',
        'sourceb' : 'k4',
        },


    'writeback'   : {
        'source' : "mem8",
        'dest'   : "rfc[rd]",
    },
}
```

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_lb"></src:fragref>
<src:fragref linkend="src_inst_impl_lb"></src:fragref>
<src:fragref linkend="src_inst_asm_lb"></src:fragref>
```

2.10. ldir1

2.10.1. Instruction definition

```
spec['useq']['inst']['ldir1'] =  {
    'fullname' : 'ldir1',
    'Description' : 'Load 12-bit immediate value in upper part of r1',
    'opcode1'      : 0xF,
    'type'        : 'Type_IX',

}
```

2.10.2. Instruction implementation

```
spec['useq']['inst']['ldir1']['impl'] =  {
    'rf' : {
        'sela' : 'r0', # read a 0
        'selb' : 'rs2', # dont care
        'selc' : 'r1', # we force the write in r1
        },

    'alu'         : {
        'op' : 'add',
        'sourcea' : 'rfa[r0]', # should be 0
        'sourceb' : 'k12',
        },
    'writeback'   : {
        'source' : 'alu',
        'dest'   : 'rfc[rd]',
    },
    'flag'        : {
        'carry_in' : 0,
        'carry'    : 'keep',
    },

}
```

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_ldir1"></src:fragref>
<src:fragref linkend="src_inst_impl_ldir1"></src:fragref>
<src:fragref linkend="src_inst_asm_ldir1"></src:fragref>
```

2.11. lw

2.11.1. Instruction definition

```
spec['useq']['inst']['lw'] =  {
    'fullname' : 'lw',
    'Description' : 'Load word',
    'opcode1'      : 3,
    'type'         : 'Type_II',

}
```

2.11.2. Instruction implementation

```
spec['useq']['inst']['lw']['impl'] =  {

    'alu'          : {
        'op' : 'add',
        'sourcea' : 'rfa[rs1]',
        'sourceb' : 'k4',
        },


    'writeback'    : {
        'source' : "mem16",
        'dest'   : "rfc[rd]",
    },
}
```

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_lw"></src:fragref>
<src:fragref linkend="src_inst_impl_lw"></src:fragref>
<src:fragref linkend="src_inst_asm_lw"></src:fragref>
```

2.12. mfsr

2.12.1. Instruction definition

```
spec['useq']['inst']['mfsr'] =  {
    'fullname' : 'mfsr',
    'Description' : 'Move from special register',
    'opcode1'      : 0xE,
    'opcode4'      : 0x1,
    'type'        : 'Type_VIII',

}
```

2.12.2. Instruction implementation

```
spec['useq']['inst']['mfsr']['impl'] =  {
    'rf' : {
        'sela' : 'r0', #  Force a zero
        'selb' : 'rs1', # # don't care
        'selc' : 'rd', #
        },

    'alu'         : {
        'op' : 'add',
        'sourcea' : 'rfa[r0]', # should be 0
        'sourceb' : 'sr[rs1]',
        },
    'writeback'   : {
        'source' : 'alu',
        'dest'   : 'rfc[rd]',
    },
    'flag'        : {
        'carry_in' : 0,
        'carry'    : 'keep',
    },

}
```

2.12.3. Assembler implementation

```
spec['useq']['inst']['mfsr']['asm'] =  {
    'main'         : '\"mfsr %reg,%sreg\"' +' ,rd,rs1',
}
```

2.12.4. Specification list

```
<src:fragref linkend="src_inst_def_mfsr"></src:fragref>
<src:fragref linkend="src_inst_impl_mfsr"></src:fragref>
<src:fragref linkend="src_inst_asm_mfsr"></src:fragref>
```

2.13. mtsr

2.13.1. Instruction definition

```
spec['useq']['inst']['mtsr'] =  {
    'fullname' : 'mtsr',
    'Description' : 'Move to special register',
    'opcode1'      : 0xE,
    'opcode4'      : 0x0,
    'type'        : 'Type_VIII',

}
```

2.13.2. Instruction implementation

```
spec['useq']['inst']['mtsr']['impl'] =  {
    'rf' : {
        'sela' : 'r0', #  Force a zero
        'selb' : 'rs1', # data source
        'selc' : 'rd', # don't care
        },

    'alu'         : {
        'op' : 'add',
        'sourcea' : 'rfa[r0]', # should be 0
        'sourceb' : 'rfb[rs1]',
        },
    'writeback'   : {
        'source' : 'alu',
        'dest'   : 'sr[rd]',
    },
    'flag'        : {
        'carry_in' : 0,
        'carry'    : 'keep',
    },

}
```

2.13.3. Assembler implementation

```
spec['useq']['inst']['mtsr']['asm'] =  {
    'main'         : '\"mtsr %sreg,%reg\"' +' ,rd,rs1',
}
```

2.13.4. Specification list

```

<src:fragref linkend="src_inst_def_mtsr"></src:fragref>
<src:fragref linkend="src_inst_impl_mtsr"></src:fragref>
<src:fragref linkend="src_inst_asm_mtsr"></src:fragref>
```

2.14. ori4

2.14.1. Instruction definition

```

spec['useq']['inst']['ori4'] =  {
    'fullname' : 'ori4',
    'Description' : 'immediate OR with 4-bit value',
    'opcode1'      : 0,
    'type'        : 'Type_I',

}
```

2.14.2. Instruction implementation

```

spec['useq']['inst']['ori4']['impl'] =  {

    'rf' : {
        'sela' : 'r0', #  yes, we just want to "move" data
        'selb' : 'rs1',
        'selc' : 'rd',
    },

    'alu'         : {
        'op' : 'or',
        'sourcea' : 'rfa[0]',
        'sourceb' : 'rfb[rs1]_or_uk4',
        },

    'flag'        : {
        'carry_in' : 0,
        'carry'    : 'keep',
    }
}
```

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_ori4"></src:fragref>
<src:fragref linkend="src_inst_impl_ori4"></src:fragref>
<src:fragref linkend="src_inst_asm_ori4"></src:fragref>
```

2.15. orr1

2.15.1. Instruction definition

```
spec['useq']['inst']['orr1'] =  {
    'fullname' : 'orr1',
    'Description' : 'Bitwise or  with r1 and  immediate value',
    'opcode1'      : 0x8,
    'opcode3'      : 0x1,
    'type'        : 'Type_IV',

}
```

2.15.2. Instruction implementation

```
spec['useq']['inst']['orr1']['impl'] =  {
    'alu'          : {
        'op' : 'or',
        'sourcea' : 'rfa[rd]',
        'sourceb' : 'rfb[r1]_or_uk4',
        },
}
```

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_orr1"></src:fragref>
<src:fragref linkend="src_inst_impl_orr1"></src:fragref>
<src:fragref linkend="src_inst_asm_orr1"></src:fragref>
```

2.16. or

2.16.1. Instruction definition

```
spec['useq']['inst']['or'] =  {
    'fullname' : 'or',
    'Description' : '16-bit bitwise or',
    'opcode1'      : 7,
    'opcode3'      : 1,
    'type'        : 'Type_III',

}
```

2.16.2. Instruction implementation

```
spec['useq']['inst']['or']['impl'] =  {

    'alu'         : {
        'op' : 'or',
        'sourcea' : 'rfa[rd]',
        'sourceb' : 'rfb[rs2]',
        },

    'flag'        : {
        'carry_in' : 0,
        'carry'    : 'keep',
    }
}
```

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

2.17. ret

2.17.1. Instruction definition

```
spec['useq']['inst']['ret'] =  {
    'fullname' : 'ret',
    'Description' : 'Return from function call',
    'opcode1'      : 0xE,
    'opcode4'      : 0x2,
    'type'         : 'Type_VIII',

}
```

2.17.2. Instruction implementation

```
spec['useq']['inst']['ret']['impl'] =  {
    'rf' : {
        'sela' : 'r0', #  Force a zero
        'selb' : 'r14', # Link register
        'selc' : 'rd', #  Don't care
        },

    'alu'          : {
        'op' : 'add',
        'sourcea' : 'rfa[r0]', # should be 0
        'sourceb' : 'rfb[r14]',
        },
    'writeback'    : {
        'source' : 'alu',
        'dest'   : 'pc',
    },
    'flag'         : {
        'carry_in' : 0,
        'carry'    : 'keep',
    },
    'pc' : {
        'next' : 'from_alu'
    },


}
```

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_ret"></src:fragref>
<src:fragref linkend="src_inst_impl_ret"></src:fragref>
<src:fragref linkend="src_inst_asm_ret"></src:fragref>
```

2.18. sb

2.18.1. Instruction definition

```
spec['useq']['inst']['sb'] =  {
    'fullname' : 'sb',
    'Description' : 'Store byte',
    'opcode1'      : 6,
    'type'        : 'Type_II',

}
```

2.18.2. Instruction implementation

```
spec['useq']['inst']['sb']['impl'] =  {

    'alu'         : {
        'op' : 'add',
        'sourcea' : 'rfa[rs1]',
        'sourceb' : 'k4',
        },


    'writeback'   : {
        'source' : "rfb[rd]",
        'dest'   : "mem8",
    },
}
```

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_sb"></src:fragref>
<src:fragref linkend="src_inst_impl_sb"></src:fragref>
<src:fragref linkend="src_inst_asm_sb"></src:fragref>
```

2.19. sext

2.19.1. Instruction definition

```
spec['useq']['inst']['sext'] =  {
    'fullname' : 'sext',
    'Description' : '8-bit to 16-bit Sign extention ',
    'opcode1'      : 7,
    'opcode3'      : 7,
    'type'         : 'Type_III',

}
```

2.19.2. Instruction implementation

```
spec['useq']['inst']['sext']['impl'] =  {

    'alu'          : {
        'op' : 'sext',
        'sourcea' : 'rfa[rd]',
        'sourceb' : 'rfb[rs2]',
        },

    'flag'         : {
        'carry_in' : 0,
        'carry'    : 'keep',
    }
}
```

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_sext"></src:fragref>
<src:fragref linkend="src_inst_impl_sext"></src:fragref>
<src:fragref linkend="src_inst_asm_sext"></src:fragref>
```

2.20. slli

2.20.1. Instruction definition

```
spec['useq']['inst']['slli'] =  {
    'fullname' : 'slli',
    'Description' : 'Shift Left, Logical, using immediate value',
    'opcode1'      : 0x8,
    'opcode3'      : 0x7,
    'type'        : 'Type_IV',

}
```

2.20.2. Instruction implementation

```
spec['useq']['inst']['slli']['impl'] =  {
    'alu'         : {
        'op' : 'sll',
        'sourcea' : 'rfa[rd]',
        'sourceb' : 'uk4',    # R1 not used
        },
    'writeback'   :   {
        'source' : "alu",
        'dest'   : "rfc[rd]",
        },

}
```

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_slli"></src:fragref>
<src:fragref linkend="src_inst_impl_slli"></src:fragref>
<src:fragref linkend="src_inst_asm_slli"></src:fragref>
```

2.21. sll

2.21.1. Instruction definition

```
spec['useq']['inst']['sll'] =  {
    'fullname' : 'sll',
    'Description' : '16-bit Shift Left, Logical ',
    'opcode1'      : 7,
    'opcode3'      : 6,
    'type'        : 'Type_III',

}
```

2.21.2. Instruction implementation

```
spec['useq']['inst']['sll']['impl'] =  {

    'alu'         : {
        'op' : 'sll',
        'sourcea' : 'rfa[rd]',
        'sourceb' : 'rfb[rs2]',
        },

    'flag'        : {
        'carry_in' : 0,
        'carry'    : 'keep',
    }
}
```

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_sll"></src:fragref>
<src:fragref linkend="src_inst_impl_sll"></src:fragref>
<src:fragref linkend="src_inst_asm_sll"></src:fragref>
```

2.22. srai

2.22.1. Instruction definition

```
spec['useq']['inst']['srai'] =  {
    'fullname' : 'srai',
    'Description' : 'Shift  Right, Arithmetic, using immediate value',
    'opcode1'      : 0x8,
    'opcode3'      : 0x5,
    'type'        : 'Type_IV',

}
```

2.22.2. Instruction implementation

```
spec['useq']['inst']['srai']['impl'] =  {
    'alu'         : {
        'op' : 'sra',
        'sourcea' : 'rfa[rd]',
        'sourceb' : 'uk4',    # R1 not used
        },
    'writeback'   :   {
        'source' : "alu",
        'dest'   : "rfc[rd]",
        },

}
```

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_srai"></src:fragref>
<src:fragref linkend="src_inst_impl_srai"></src:fragref>
<src:fragref linkend="src_inst_asm_srai"></src:fragref>
```

2.23. sra

2.23.1. Instruction definition

```
spec['useq']['inst']['sra'] =  {
    'fullname' : 'sra',
    'Description' : '16-bit Shift Right, arithmetic',
    'opcode1'      : 7,
    'opcode3'      : 4,
    'type'        : 'Type_III',

}
```

2.23.2. Instruction implementation

```
spec['useq']['inst']['sra']['impl'] =  {

    'alu'         : {
        'op' : 'sra',
        'sourcea' : 'rfa[rd]',
        'sourceb' : 'rfb[rs2]',
        },

    'flag'        : {
        'carry_in' : 0,
        'carry'    : 'keep',
    }
}
```

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_sra"></src:fragref>
<src:fragref linkend="src_inst_impl_sra"></src:fragref>
<src:fragref linkend="src_inst_asm_sra"></src:fragref>
```

2.24. srli

2.24.1. Instruction definition

```
spec['useq']['inst']['srli'] =  {
    'fullname' : 'srli',
    'Description' : 'Shift Right, Logical, using immediate value',
    'opcode1'      : 0x8,
    'opcode3'      : 0x6,
    'type'        : 'Type_IV',

}
```

2.24.2. Instruction implementation

```
spec['useq']['inst']['srli']['impl'] =  {
    'alu'         : {
        'op' : 'srl',
        'sourcea' : 'rfa[rd]',
        'sourceb' : 'uk4',    # R1 not used
        },
    'writeback'   :   {
        'source' : "alu",
        'dest'   : "rfc[rd]",
        },

}
```

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_srli"></src:fragref>
<src:fragref linkend="src_inst_impl_srli"></src:fragref>
<src:fragref linkend="src_inst_asm_srli"></src:fragref>
```

2.25. srl

2.25.1. Instruction definition

```
spec['useq']['inst']['srl'] =  {
    'fullname' : 'srl',
    'Description' : '16-bit Shift Right, Logical ',
    'opcode1'      : 7,
    'opcode3'      : 5,
    'type'        : 'Type_III',

}
```

2.25.2. Instruction implementation

```
spec['useq']['inst']['srl']['impl'] =  {

    'alu'         : {
        'op' : 'srl',
        'sourcea' : 'rfa[rd]',
        'sourceb' : 'rfb[rs2]',
        },

    'flag'        : {
        'carry_in' : 0,
        'carry'    : 'keep',
    }
}
```

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_srl"></src:fragref>
<src:fragref linkend="src_inst_impl_srl"></src:fragref>
<src:fragref linkend="src_inst_asm_srl"></src:fragref>
```

2.26. sub2c

2.26.1. Instruction definition

```
spec['useq']['inst']['sub2c'] =  {
    'fullname' : 'sub2c',
    'Description' : '16-bit substraction with carry ',
    'opcode1'      : 7,
    'opcode3'      : 9,
    'type'        : 'Type_III',

}
```

2.26.2. Instruction implementation

```
spec['useq']['inst']['sub2c']['impl'] =  {

    'alu'         : {
        'op' : 'add',
        'sourcea' : 'rfa[rd]',
        'sourceb' : 'not_rfb[rs2]',
        },

    'flag'        : {
        'carry_in' : 'not_carry',
        'carry'    : 'update',
    }
}
```

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_sub2c"></src:fragref>
<src:fragref linkend="src_inst_impl_sub2c"></src:fragref>
<src:fragref linkend="src_inst_asm_sub2c"></src:fragref>
```

2.27. sub2

2.27.1. Instruction definition

```
spec['useq']['inst']['sub2'] =  {
    'fullname' : 'sub2',
    'Description' : '16-bit signed substraction',
    'opcode1'      : 2,
    'type'        : 'Type_I',

}
```

2.27.2. Instruction implementation

```
spec['useq']['inst']['sub2']['impl'] =  {

    'alu'         : {
        'op' : 'add',
        'sourcea' : 'rfa[rd]',
        'sourceb' : 'not_rfb[rs1]_or_uk4',
        },

    'flag'        : {
        'carry_in' : 1,
        'carry'    : 'update',
    }
}
```

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_sub2"></src:fragref>
<src:fragref linkend="src_inst_impl_sub2"></src:fragref>
<src:fragref linkend="src_inst_asm_sub2"></src:fragref>
```

2.28. sub3

2.28.1. Instruction definition

```
spec['useq']['inst']['sub3'] =  {
    'fullname' : 'sub3',
    'Description' : '16-bit signed addition',
    'opcode1'      : 0xD,
    'type'        : 'Type_VII',

}
```

2.28.2. Instruction implementation

```
spec['useq']['inst']['sub3']['impl'] =  {

    'alu'         : {
        'op' : 'sub',
        'sourcea' : 'rfa[rs1]',
        'sourceb' : 'rfb[rs2]',
        },
    'flag'        : { # some type_IV instructions will override that
        'carry_in' : 1,
        'carry'    : 'update',
    },

}
```

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_sub3"></src:fragref>
<src:fragref linkend="src_inst_impl_sub3"></src:fragref>
<src:fragref linkend="src_inst_asm_sub3"></src:fragref>
```

2.29. sw

2.29.1. Instruction definition

```
spec['useq']['inst']['sw'] =  {
    'fullname' : 'sw',
    'Description' : 'Store word',
    'opcode1'      : 5,
    'type'        : 'Type_II',

}
```

2.29.2. Instruction implementation

```
spec['useq']['inst']['sw']['impl'] =  {

    'alu'         : {
        'op' : 'add',
        'sourcea' : 'rfa[rs1]',
        'sourceb' : 'k4',
        },


    'writeback'   : {
        'source' : "rfb[rd]",
        'dest'   : "mem16",
    },
}
```

2.29.3. Assembler implementation

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

2.29.4. Specification list

```
<src:fragref linkend="src_inst_def_sw"></src:fragref>
<src:fragref linkend="src_inst_impl_sw"></src:fragref>
<src:fragref linkend="src_inst_asm_sw"></src:fragref>
```

2.30. xnorr1

2.30.1. Instruction definition

```
spec['useq']['inst']['xnorr1'] =  {
    'fullname' : 'xnorr1',
    'Description' : 'Bitwise xnor  with r1 and  immediate value',
    'opcode1'      : 0x8,
    'opcode3'      : 0x3,
    'type'        : 'Type_IV',

}
```

2.30.2. Instruction implementation

```
spec['useq']['inst']['xnorr1']['impl'] =  {
    'alu'         : {
        'op' : 'xnor',
        'sourcea' : 'rfa[rd]',
        'sourceb' : 'rfb[r1]_or_uk4',
        },
}
```

2.30.3. Assembler implementation

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

2.30.4. Specification list

```
<src:fragref linkend="src_inst_def_xnorr1"></src:fragref>
<src:fragref linkend="src_inst_impl_xnorr1"></src:fragref>
<src:fragref linkend="src_inst_asm_xnorr1"></src:fragref>
```

2.31. xnor

2.31.1. Instruction definition

```
spec['useq']['inst']['xnor'] =  {
    'fullname' : 'xnor',
    'Description' : '16-bit bitwise negated exclusive or',
    'opcode1'      : 7,
    'opcode3'      : 3,
    'type'        : 'Type_III',

}
```

2.31.2. Instruction implementation

```
spec['useq']['inst']['xnor']['impl'] =  {

    'alu'         : {
        'op' : 'xnor',
        'sourcea' : 'rfa[rd]',
        'sourceb' : 'rfb[rs2]',
        },

    'flag'        : {
        'carry_in' : 0,
        'carry'    : 'keep',
    }
}
```

2.31.3. Assembler implementation

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

2.31.4. Specification list

```
<src:fragref linkend="src_inst_def_xnor"></src:fragref>
<src:fragref linkend="src_inst_impl_xnor"></src:fragref>
<src:fragref linkend="src_inst_asm_xnor"></src:fragref>
```

2.32. xorr1

2.32.1. Instruction definition

```
spec['useq']['inst']['xorr1'] =  {
    'fullname' : 'xorr1',
    'Description' : 'Bitwise xor  with r1 and  immediate value',
    'opcode1'      : 0x8,
    'opcode3'      : 0x2,
    'type'        : 'Type_IV',
}
```

2.32.2. Instruction implementation

```
spec['useq']['inst']['xorr1']['impl'] =  {
    'alu'          : {
        'op' : 'xor',
        'sourcea' : 'rfa[rd]',
        'sourceb' : 'rfb[r1]_or_uk4',
        },
}
```

2.32.3. Assembler implementation

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

2.32.4. Specification list

```
<src:fragref linkend="src_inst_def_xorr1"></src:fragref>
<src:fragref linkend="src_inst_impl_xorr1"></src:fragref>
<src:fragref linkend="src_inst_asm_xorr1"></src:fragref>
```

2.33. xor

2.33.1. Instruction definition

```
spec['useq']['inst']['xor'] =  {
    'fullname' : 'xor',
    'Description' : '16-bit bitwise exclusive or',
    'opcode1'      : 7,
    'opcode3'      : 2,
    'type'        : 'Type_III',

}
```

2.33.2. Instruction implementation

```
spec['useq']['inst']['xor']['impl'] =  {
    'alu'         : {
        'op' : 'xor',
        'sourcea' : 'rfa[rd]',
        'sourceb' : 'rfb[rs2]',
        },

    'flag'        : {
        'carry_in' : 0,
        'carry'    : 'keep',
    }
}
```

2.33.3. Assembler implementation

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

2.33.4. 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>
```

Internal GAS Pseudo-instructions

Pseudo-instructions (or synthetic instructions) are defined using native instructions. GAS can be configured through Archc to recognize those instructions directly and output corresponding native instructions. This simplifies assembly code writing and C compiler retargeting.

Important

The Archc code for pseudo-instructions recognized by GAS is generated automatically generated from the Python spec file. Don't edit Archc file directly !

3.1. zldi

3.1.1. Instruction definition

```
spec['useq']['pseudo_inst']['zldi'] =  {
    'fullname' : 'zldi',
    'Description' : 'Load Immediate',
    'fields'       : '%reg,%exp',
    'code' : """
        "ldir1  (%1>>4)       ";
        "ori4   %0,r1,(%1 & 0x0F) ";
"""
}
```

3.1.2. Specification list

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

3.2. zjump

3.2.1. Instruction definition

```
spec['useq']['pseudo_inst']['zjump'] =  {
    'fullname' : 'zjump',
    'Description' : 'Jump to an immediate address ',
    'fields'       : '%exp',
    'code' : """
        "ldir1  (%0>>4)       ";
        "br   r1,(%0 & 0x0F)";
"""
}
```

3.2.2. Specification list

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

Pseudo-instructions implemented as GAS macro

Some instructions can't be implement directly in GAS using Archc. Therefore, they are implemented as GAS macros using the .macro .endm directives

4.1. 32-bit arithmetic instruction

4.1.1. zadd32 : 32-bit addition

Used to implement 32-bit addition for the C compiler

```
;; fixme - no carry handling....
.macro  zadd32 src1,src2,dest
        add3 \dest,\src1,\src2
        add3 (\dest+1),(\src1+1),(\src2+1)
.endm
```

```
<src:fragref linkend="src_inst_zadd32"></src:fragref>
```

Processing Element detailled instruction set

Note to the reader

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

5.1. Instruction formats

5.1.1. Type I

5.1.1.1. Format definition

```
spec['pe']['inst_type']['Type_I']['format']= {
    'mem' : {'size'   : 2,
              'offset' : 24,
          },

    'agen' : {'size'   : 4,
              'offset' : 20,
          },

    'cond' : {'size'   : 4,
               'offset' : 16,
                },
    'aluop' : {'size'   : 4,
               'offset' : 12,
               'decode' : True,  # Field is used for decoding
                },
    'rd'     : {'size'   : 4,
                'offset' : 8
                },
    'rs1'     : {'size'   : 4,
                'offset'  : 4
                },
    'rs2'     : {'size'   : 4,
                'offset'  : 0
                },
}



# default implementation
spec['pe']['inst_type']['Type_I']['impl']= {
    'rf' : {
        'sela' : 'rs1',
        'selb' : 'rs2',
        'selc' : 'rd',
    },

    'writeback'   : {
        'source' : "alu",
        'dest'   : "rfc[rd]",
    },
}

spec['useq']['inst_type']['Type_I']['asm']   =  '%reg, %reg, %exp';
spec['useq']['inst_type']['Type_I']['fields']=  'rd, rs1, uk4';
# just an helper for quick docbook generation - we could write some code to do it
 automatically...
spec['useq']['inst_type']['Type_I']['docbook']=  ('mem','agen', 'cond',
 'aluop','rd','rs1','rs2');
```

5.1.1.2. Specification list

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

5.2. p_add

5.2.1. Instruction definition

```
spec['pe']['inst']['p_add'] =  {
    'fullname' : 'p_add',
    'Description' : 'signed addition',
    'aluop'      : 0x0,
    'type'        : 'Type_I',

}
```

5.2.2. Instruction implementation

```
spec['pe']['inst']['p_add']['impl'] =  {

    'alu'         : {
        'op' : 'add',
        'sourcea' : 'rfa[rs1]',
        'sourceb' : 'rfb[rs2]',
        },
    'flag'        : {
        'carry_in' : 0,
        'carry'    : 'update',
    },

}
```

5.2.3. Assembler implementation

```
# Special variant to support expression instead of "pure" register name
# - To be used in gas macro
spec['pe']['inst']['p_add']['asm'] =  {
    'main'          : '\"p_add %reg,%reg,%reg\"' +' ,rd,rs1,rs2',
    'variant1'      : '\"p_add %exp,%exp,%exp\"' + ',rd,rs1,rs2',
}
```

5.2.4. Specification list

```
<src:fragref linkend="src_pe_inst_def_p_add"></src:fragref>
<src:fragref linkend="src_pe_inst_impl_p_add"></src:fragref>
<src:fragref linkend="src_pe_inst_asm_p_add"></src:fragref>
```

5.3. p_or

5.3.1. Instruction definition

```
spec['pe']['inst']['p_or'] =  {
    'fullname' : 'p_or',
    'Description' : 'Bitwise OR',
    'aluop'      : 0x1,
    'type'        : 'Type_I',

}
```

5.3.2. Instruction implementation

```
spec['pe']['inst']['p_or']['impl'] =  {

    'alu'         : {
        'op' : 'or',
        'sourcea' : 'rfa[rs1]',
        'sourceb' : 'rfb[rs2]',
        },
    'flag'        : {
        'carry_in' : 0,
        'carry'    : 'keep',
    },

}
```

5.3.3. Assembler implementation

```
# Special variant to support expression instead of "pure" register name
# - To be used in gas macro
spec['pe']['inst']['p_or']['asm'] =  {
    'main'         : '\"p_or %reg,%reg,%reg\"' +' ,rd,rs1,rs2',
    'variant1'     : '\"p_or %exp,%exp,%exp\"' + ',rd,rs1,rs2',
}
```

5.3.4. Specification list

```
<src:fragref linkend="src_pe_inst_def_p_or"></src:fragref>
<src:fragref linkend="src_pe_inst_impl_p_or"></src:fragref>
<src:fragref linkend="src_pe_inst_asm_p_or"></src:fragref>
```

Appendix A. Revision History

Revision 0-0	**Fri Sep 14 2012**	**Ronan Barzic** *ronan.barzic@atmel.comm*
Initial creation		

Index