Zuerst werden die arithmetischen Immediate Befehle getestet, da diese in sich abgeschlossen sind.
Sind diese Befehle erfolgreich getestet, so können mit deren Hilfe die Register nutzenden arithmetischen Befehle getestet werden.

Nachdem alle arithmetischen Befehle getestet wurden, können die set Befehle getestet werden. Da diese bereits die Vergleichshardware nutzen, aber noch keinen Sprung auslösen.

Anschließend werden die Sprungbefehle getestet um die Sprungeinheiten im Steuerwerk zu testen.

Wenn Sprünge und Set Befehle funktionieren, können die Branches getestet werden da die Vergleichshardware und die Sprunghardware bereits funktioniert.

Nun zu etwas völlig anderem: Lade/Speicherbefehle.
Zuerst werden die Load Befehle getestet im ROM Bereich.
Danach werden die Speicherbefehle getestet und in den Speicher Schreibetests wird auch in den RAM Bereich geschrieben und danach wieder ausgelesen.

Jetzt könne die Exceptions getestet werden und dessen Handling. Gleichzeitig dazu werden RFE und COP2 (EPC auslesen) damit getestet.

Für die Krönung (mul/div) fehlt nun nurnoch das Lesen/beschreiben der HI/LO Register.

s = Step getestet (Mikrocode für Mikrocode)
c = Command Step getestet (Befehl für Befehl)
f = free Running getestet (einmal durchlaufen lassen bis zur Endlosschleife am Ende)

Immediate Befehle

lui:
Das Programm läuft von 0x00000000 bis 0x00000044

```
main:
        lui $1, 1
        lui $2, 2
        lui $3, 4
        lui $4, 8
        lui $5, 16
        lui $6, 32
        lui $7, 64
        lui $8, 128
        lui $9, 256
        lui $10, 512
        lui $11, 1024
        lui $12, 2048
        lui $13, 4096
        lui $14, 8192
        lui $15, 16384
        lui $16, 32768
idle:
        j idle
```

Danach müssen die Register so aussehen:

Register	Inhalt	OK
1	0x0001 0000	scf
2	0x0002 0000	Scf
3	0x0004 0000	Scf
4	0x0008 0000	Scf
5	0x0010 0000	Scf
6	0x0020 0000	Scf
7	0x0040 0000	Scf
8	0x0080 0000	Scf
9	0x0100 0000	Scf
10	0x0200 0000	Scf
11	0x0400 0000	Scf
12	0x0800 0000	Scf
13	0x1000 0000	Scf
14	0x2000 0000	Scf
15	0x4000 0000	Scf
16	0x8000 0000	Scf

addi:
Das Programm läuft von 0x00000000 bis 0x00000030

```
main:
        addi $1, $0, 0x32
        addi $2, $1, 0x1
        addi $3, $2, 0x1
        addi $4, $3, 0x1
        addi $5, $4, 0x1
        addi $6, $5, 0x1
        addi $7, $6, 0x1
        addi $8, $7, 0x1
        addi $9, $8, 0xffff
        lui $10, 0x8000
        addi $10, $10, -1
idle:
        j idle
```

Danach müssen die Register so aussehen:

Register	Inhalt	OK
1	0x32	scf
2	0x33	Scf
3	0x34	Scf
4	0x35	Scf
5	0x36	Scf
6	0x37	Scf
7	0x38	Scf
8	0x39	Scf
9	0x38	Scf
10	0x80000000 (EXC1)	Scf

addiu:
Das Programm läuft von 0x00000000 bis 0x00000038

```
main:
        addiu $1, $0, 0x32
        addiu $2, $1, 0x1
        addiu $3, $2, 0x1
        addiu $4, $3, 0x1
        addiu $5, $4, 0x1
        addiu $6, $5, 0x1
        addiu $7, $6, 0x1
        addiu $8, $7, 0x1
        addiu $9, $8, 0xffff
        lui   $10, 0x8000
        addiu $10, $10, -1
        addiu $30, $9, 0xffff
        addiu $31, $30, 0xffff
idle:
        j idle
```

Danach müssen die Register so aussehen

Register	Inhalt	OK
1	0x32	scf
2	0x33	Scf
3	0x34	Scf
4	0x35	Scf
5	0x36	Scf
6	0x37	Scf
7	0x38	Scf
8	0x39	Scf
9	0x38	Scf
10	0x7FFFFFFF	Scf
31	0x36	Scf

ori:
Das Programm läuft von 0x00000000 bis 0x00000068

```
main:
        ori $1, $0, 0x64
        ori $2, $1, 0x80
        ori $3, $0, 0x64
        ori $4, $0, 0x64
        ori $5, $0, 0x64
        ori $6, $0, 0x64
        ori $7, $0, 0x64
        ori $8, $0, 0x64
        ori $9, $0, 0x64
        ori $10, $0, 0x64
        ori $11, $0, 0x64
        ori $12, $0, 0x64
        ori $13, $0, 0x64
        ori $14, $0, 0x64
        ori $15, $0, 0x64
        ori $16, $0, 0x64
        ori $17, $0, 0x64
        ori $18, $0, 0x64
        ori $19, $0, 0x64
        ori $20, $0, 0x64
        ori $21, $0, 0x64
        ori $22, $0, 0x64
        ori $23, $0, 0x64
        ori $24, $0, 0x64
        ori $25, $0, 0x64
idle:
        j idle
```

Danach müssen die Register so aussehen

Register	Inhalt	OK
1	0x64	scf
2	0xE4	Scf
3	0x64	Scf
4	0x64	Scf
5	0x64	Scf
6	0x64	Scf
7	0x64	Scf
8	0x64	Scf
9	0x64	Scf
10	0x64	Scf
11	0x64	Scf
12	0x64	Scf
13	0x64	Scf
14	0x64	Scf
15	0x64	Scf
16	0x64	Scf
17	0x64	Scf

18	0x64	Scf
19	0x64	Scf
20	0x64	Scf
21	0x64	Scf
22	0x64	Scf
23	0x64	Scf
24	0x64	Scf
25	0x64	Scf

andi:
Das Programm läuft von 0x00000000 bis 0x00000014

```
main:
        andi $1, $0, 0xffff
        ori  $2, $1, 0x0f0f
        andi $3, $2, 0xf0f0
        andi $4, $2, 0x0505
idle:
        j idle
```

Danach müssen die Register so aussehen

Register	Inhalt	OK
1	0x0	Scf
2	0x0F0F	Scf
3	0x0	Scf
4	0x0505	Scf

xori:
Das Programm läuft von 0x00000000 bis 0x00000010

```
main:
        ori  $1, $0, 0x5555
        xori $2, $1, 0xaaaa
        xori $3, $2, 0xffff
idle:
        j idle
```

Danach müssen die Register so aussehen

Register	Inhalt	OK
1	0x5555	Scf
2	0xFFFF	Scf
3	0x0	scf

Arithmetische Befehle

add:

```
main:
        ori $1, $0, 0x4fff
        ori $2, $0, 0x0007
        add $3, $2, $1
        lui $4, 0xffff
        ori $4, $4, 0xffff
        add $5, $1, $4
        #overflow
        lui $6, 0x8000
        add $8, $6, $4
idle:
        j idle
```

Register	Inhalt	OK
1	0x4FFF	scf
2	0x7	Scf
3	0x5006	Scf
4	0xFFFF FFFF	Scf
5	0x4FFE (Overflow)	Scf
6	0x8000 0000	Scf
8	Vorheriger Inhalt (EXC1)	Scf

addu:

```
main:
        ori $1, $0, 0x4fff
        ori $2, $0, 0x0007
        addu $3, $2, $1
        lui $4, 0xffff
        ori $4, $4, 0xffff
        addu $5, $1, $4
        #overflow
        lui $6, 0x8000
        addu $8, $6, $4
idle:
        j idle
```

Register	Inhalt	OK
1	0x4FFF	scf
2	0x7	Scf
3	0x5006	Scf
4	0xFFFF FFFF	Scf
5	0x4FFE	Scf
6	0x8000 0000	Scf
8	0x7FFF FFFF	Scf

and:

```
main:
        ori $1, $0, 0x5555
        ori $2, $0, 0xaaaa
        ori $3, $0, 0x5555
        and $4, $1, $2
        and $5, $1, $3
        lui $6, 0x5555
        ori $6, $6, 0xaaaa
        lui $7, 0x5555
        ori $7, $7, 0xaaaa
        and $20, $7, $6
        and $21, $6, $7
        and $22, $2, $7
idle:
        j idle
```

Register	Inhalt	OK
1	0x5555	scf
2	0xAAAA	Scf
3	0x5555	Scf
4	0x0	Scf
5	0x5555	Scf
6	0x5555 AAAA	Scf
7	0x5555 AAAA	Scf
20	0x5555 AAAA	Scf
21	0x5555 AAAA	Scf
22	0xAAAA	Scf

nor:

```
main:
        ori $1, $0, 0x5555
        ori $2, $0, 0xaaaa
        nor $3, $2, $1
        nor $4, $1, $2
        lui $5, 0x7777
        nor $6, $4, $3
idle:
        j idle
```

Register	Inhalt	OK
1	0x5555	scf
2	0xAAAA	Scf
3	0xFFFF 0000	Scf
4	0xFFFF 0000	Scf
5	0x7777 0000	Scf
6	0xFFFF	Scf

or:

```
main:
        ori $1, $0, 0x5555
        ori $2, $0, 0xaaaa
        or  $3, $2, $1
        or  $4, $1, $2
        lui $5, 0x7777
        or  $6, $5, $3
idle:
        j idle
```

Register	Inhalt	OK
1	0x5555	scf
2	0xAAAA	Scf
3	0xFFFF	Scf
4	0xFFFF	Scf
5	0x7777 0000	Scf
6	0x7777 FFFF	Scf

sll:

```
main:
        ori $1, $0, 0x0001
        sll $2, $1, 15
        sll $3, $1, 16
        sll $4, $1, 31
idle:
        j idle
```

Register	Inhalt	OK
1	0x1	scf
2	0x8000	Scf
3	0x1 0000	Scf
4	0x 8000 0000	scf

sllv:

```
main:
        ori $1, $0, 0x0001
        ori $2, $0, 15
        ori $3, $0, 16
        ori $4, $0, 31
        sllv $5, $1, $2
        sllv $6, $1, $3
        sllv $7, $1, $4
idle:
        j idle
```

Register	Inhalt	OK
1	0x1	scf
2	0xF	Scf
3	0x10	Scf
4	0x1F	Scf
5	0x8000	Scf
6	0x1 0000	Scf
7	0x8000 0000	Scf

sra:

```
main:
        lui $1, 0x8000
        sra $2, $1, 15
        sra $3, $1, 16
        sra $4, $1, 31
        lui $8, 0x4000
        sra $9, $8, 15
        sra $10, $8, 16
        sra $11, $8, 31
idle:
        j idle
```

Register	Inhalt	OK
1	0x8000 0000	scf
2	0xFFFF 0000	Scf
3	0xFFFF 8000	Scf
4	0xFFFF FFFF	Scf
8	0x4000 0000	Scf
9	0x0000 8000	Scf
10	0x0000 4000	Scf
11	0x0000 0000	scf

srlv:

main:

```
lui $1, 0x8000
ori $2, $0, 15
ori $3, $0, 16
ori $4, $0, 31
srav $5, $1, $2
srav $6, $1, $3
srav $7, $1, $4
lui $10, 0x4000
srav $11, $10, $2
srav $12, $10, $3
srav $13, $10, $4
```

idle:

```
j idle
```

Register	Inhalt	OK
1	0x8000 0000	scf
2	0xF	Scf
3	0x10	Scf
4	0x1F	Scf
5	0xFFFF 0000	scf
6	0xFFFF 8000	Scf
7	0xFFFF FFFF	Scf
10	0x4000 0000	Scf
11	0x0000 8000	Scf
12	0x0000 4000	Scf
13	0x0000 0000	Scf

srl:

```
main:
        lui $1, 0x8000
        srl $2, $1, 15
        srl $3, $1, 16
        srl $4, $1, 31
idle:
        j idle
```

Register	Inhalt	OK
1	0x8000 0000	scf
2	0x0001 0000	Scf
3	0x0000 8000	Scf
4	0x0000 0001	Scf

srlv:

main:
    lui $1, 0x8000
    ori $2, $0, 15
    ori $3, $0, 16
    ori $4, $0, 31
    srlv $5, $1, $2
    srlv $6, $1, $3
    srlv $7, $1, $4
idle:
    j idle

Register	Inhalt	OK
1	0x8000 0000	scf
2	0xF	Scf
3	0x10	Scf
4	0x1F	Scf
5	0x0001 0000	Scf
6	0x0000 8000	Scf
7	0x0000 0001	Scf

sub:

main:
```
    ori $1, $0, 0x4fff
    ori $2, $0, 0x0007
    sub $3, $1, $2
    lui $4, 0xffff
    ori $4, $4, 0xffff
    sub $5, $1, $4
    ori $6, $0, 0x0001
    sub $7, $1, $6
    sub $8, $6, $1 #kein overflow!
    li $10, 0x80000000
    sub $11, $0, $10 #overflow!
```
idle:
```
    j idle
```

Register	Inhalt	OK
1	0x4FFF	scf
2	0x7	Scf
3	0x4FF8	Scf
4	0xFFFF FFFF	Scf
5	0x5000	Scf
6	0x1	Scf
7	0x4FFE	Scf
8	0xFFFF B002	Scf
10	0x8000 0000	Scf
11	Vorheriger Inhalt (EXC1)	scf

subu:

```
main:
        ori $1, $0, 0x4fff
        ori $2, $0, 0x0007
        subu $3, $1, $2
        lui $4, 0xffff
        ori $4, $4, 0xffff
        subu $5, $1, $4
        ori $6, $0, 0x0001
        subu $7, $1, $6
        subu $8, $6, $1 #kein overflow!
        li $10, 0x80000000
        subu $11, $0, $10 #overflow!
        subu $12, $10, $6 #overflow!
idle:
        j idle
```

Register	Inhalt	OK
1	0x4FFF	scf
2	0x7	Scf
3	0x4FF8	Scf
4	0xFFFF FFFF	Scf
5	0x5000	Scf
6	0x1	Scf
7	0x4FFE	Scf
8	0xFFFF B002	Scf
10	0x8000 0000	Scf
11	0x8000 0000	Scf
12	0x7FFF FFFF	Scf

```
xor:
main:
        ori $1, $0, 0x5555
        ori $2, $0, 0xaaaa
        xor $3, $2, $1
        xor $4, $1, $2
        lui $5, 0x7777
        xor $6, $4, $3
idle:
        j idle
```

Register	Inhalt	OK
1	0x5555	scf
2	0xAAAA	Scf
3	0xFFFF	Scf
4	0xFFFF	Scf
5	0x7777 0000	Scf
6	0x0	scf

Set Befehle

slt:

main:
```
        addi $1, $0, -1
        slt $10, $1, $0
        slt $11, $0, $1
        slt $12, $0, $0
idle:
        j idle
```

Register	Inhalt	OK
1	0xFFFF FFFF	Scf
10	0x1	Scf
11	0x0	Scf
12	0x0	Scf

sltu:

```
main:
        addi $1, $0, -1
        sltu $10, $1, $0
        sltu $11, $0, $1
        sltu $12, $0, $0
idle:
        j idle
```

Register	Inhalt	OK
1	0xFFFF FFFF	Scf
10	0x0	Scf
11	0x1	Scf
12	0x0	scf

slti:

```
main:
        addi $1, $0, -1
        slti $10, $1, 0
        slti $11, $0, -1
        slti $12, $0, 0
idle:
        j idle
```

Register	Inhalt	OK
1	0xFFFF FFFF	Scf
10	0x1	Scf
11	0x0	Scf
12	0x0	Scf

sltiu:

```
main:
        addi $1, $0, -1
        sltiu $10, $1, 0
        sltiu $11, $0, -1
        sltiu $12, $0, 0
idle:
        j idle
```

Register	Inhalt	OK
1	0xFFFF FFFF	Scf
10	0x0	Scf
11	0x1	Scf
12	0x0	Scf

Sprungbefehle

j:

```
main:
        lui $1, 0xffff
        ori $1, $1, 0xfff4
        ori $2, $0, 0x42
        j labeled
        addiu $3, $2, -1
        nop
        nop
        nop
        addu $4, $1, $2

labeled:
        addi $10, $0, 0xabcd
idle:
        j idle
```

Register	Inhalt	OK
1	0xFFFF FFF4	Scf
2	0x42	Scf
3	Vorheriger Inhalt	Scf
4	Vorheriger Inhalt	Scf
10	0xFFFF ABCD	Scf

jal/jr:

main:
    lui $1, 0xffff
    ori $1, $1, 0xfff4
    ori $2, $0, 0x42
    jal labeled
    addiu $3, $2, -1 (0x0000 0014)
    nop
    nop
    nop
    addu $4, $1, $2
idle: 0x 00000028
    j idle

labeled: ox 00000030
    addi $10, $0, 0xabcd
    jr $31

Register	Inhalt	OK
1	0xFFFF FFF4	Scf
2	0x42	Scf
3	0x41	Scf
4	0x36	Scf
10	0xFFFF ABCD	Scf
31	0x14	Scf

jalr:

main:
    lui $1, 0xffff
    ori $1, $1, 0xfff4
    ori $2, $0, 0x42
    la $10, labeled
    jalr $11, $10
    addiu $3, $2, -1 (0x0000 001C)
    nop
    nop
    nop
    addu $15, $1, $2
idle:
    j idle

labeled: (0x0000 0038)
    addi $5, $0, 0xabcd
    jr $11

Register	Inhalt	OK
1	0xFFFF FFF4	Scf
2	0x42	Scf
3	0x41	Scf
5	0xFFFF ABCD	Scf
10	0x0000 0038	Scf
11	0x0000 001C	Scf
15	0x36	Scf
31	Unverändert	Scf

Branches

beq:

```
main:
        lui $1, 0xffff
        ori $1, $1, 0xfff4
        ori $2, $0, 0x42
        beq $0, $1 idle2
        beq $0, $0, labeled
        addu $10, $1, $2
        nop
        nop
        nop
        addu $11, $1, $2
idle2:
        j idle2

labeled:
        addi $12, $0, 0xabcd
idle:
        j idle
```

Register	Inhalt	OK
1	0xFFFF FFF4	Scf
2	0x42	Scf
10	Unverändert	Scf
11	Unverändert	Scf
12	0xFFFF ABCD	scf

bgez:

```
main:
        lui $1, 0xffff
        ori $1, $1, 0xfff4
        ori $2, $0, 0x42
        bgez $1, idle2
        bgez $2, labeled
        addu $3, $1, $2
        nop
        nop
        nop
        addu $4, $1, $2
idle2:
        j idle2

labeled:
        addi $10, $0, 0xabcd
        bgez $0, labeled
idle:
        j idle
```

Register	Inhalt	OK
1	0xFFFF FFF4	Scf
2	0x42	Scf
3	Unverändert	Scf
4	Unverändert	Scf
10	0xFFFF ABCD	Scf

bgezal:

```
main:
        lui $1, 0xffff
        ori $1, $1, 0xfff4
        ori $2, $0, 0x42
        bgezal $1, idle
        bgezal $2, labeled
        addiu $3, $2, -1
        nop
        nop
        nop
        addu $4, $1, $2
idle:
        bgezal $0, idle

labeled:
        addi $10, $0, 0xabcd
        jr $31
```

Register	Inhalt	OK
1	0xFFFF FFF4	Scf
2	0x42	Scf
3	0x41	Scf
4	0x36	Scf
10	0x FFFF ABCD	Scf
31	0x1C und dann 0x38	Scf

bgtz:

```
main:
        lui $1, 0xffff
        ori $1, $1, 0xfff4
        ori $2, $0, 0x42
        bgtz $0, idle2
        bgtz $1, idle2
        bgtz $2, labeled
        addu $3, $1, $2
        nop
        nop
        addu $4, $1, $2
idle2:
        addi $10, $0, 0xffff
        j idle2

labeled:
        addi $10, $0, 0xabcd
idle:
        j idle
```

Register	Inhalt	OK
1	0xFFFF FFF4	Scf
2	0x42	Scf
3	Unverändert	Scf
4	Unverändert	Scf
10	0xFFFF ABCD	Scf

blez:

```
main:
        lui $1, 0xffff
        ori $1, $1, 0xfff4
        ori $2, $0, 0x42
        blez $2, idle
        blez $1, labeled
        addu $3, $1, $2
        nop
        nop
        addu $4, $1, $2
idle:
        addi $10, $0, 0xffff
        j idle

labeled:
        addi $10, $0, 0xabcd
        blez $0, labeled
```

Register	Inhalt	OK
1	0xFFFF FFF4	Scf
2	0x42	Scf
3	Unverändert	Scf
4	Unverändert	Scf
10	0xFFFF ABCD	Scf

bltz:

```
main:
        lui $1, 0xffff
        ori $1, $1, 0xfff4
        ori $2, $0, 0x42
        bltz $0, idle2
        bltz $2, idle2
        bltz $1, labeled
        addu $3, $1, $2
        nop
        nop
        addu $4, $1, $2
idle2:
        addi $10, $0, 0xffff
        j idle2

labeled:
        addi $10, $0, 0xabcd
        j labeled
```

Register	Inhalt	OK
1	0xFFFF FFF4	Scf
2	0x42	Scf
3	Unverändert	Scf
4	Unverändert	Scf
10	0xFFFF ABCD	Scf

bltzal:

```
main:
        lui $1, 0xffff
        ori $1, $1, 0xfff4
        ori $2, $0, 0x42
        bltzal $0, idle2
        bltzal $2, idle2
        bltzal $1, labeled
        addiu $3, $2, -1
        nop
        nop
        nop
        addu $4, $1, $2
idle2:
        j idle2

labeled:
        addi $10, $0, 0xabcd
        jr $31
```

Register	Inhalt	OK
1	0xFFFF FFF4	Scf
2	0x42	Scf
3	0x41	Scf
4	0x36	Scf
10	0xFFFF ABCD	Scf
31	0x0000 0024	Scf

bne:

```
main:
        lui $1, 0xffff
        ori $1, $1, 0xfff4
        ori $2, $0, 0x42
        bne $0, $0, idle2
        bne $2, $1, labeled
        addu $3, $1, $2
        nop
        nop
        addu $4, $1, $2
idle2:
        addi $10, $0, 0xffff
        j idle2

labeled:
        addi $10, $0, 0xabcd
        j labeled
```

Register	Inhalt	OK
1	0xFFFF FFF4	Scf
2	0x42	Scf
3	Unverändert	Scf
4	Unverändert	Scf
10	0xFFFF ABCD	Scf

Speicherbefehle

lw:

```
main:
        la $1, var #load address
        lw $4, var
idle:
        j idle

.text
var: .word 0x80818283
```

Register	Inhalt	OK
1	0x0000 0018	Scf
4	0x8081 8283	Scf

lhu:

```
main:
        la $1, var #load address
        lhu $4, var
        lhu $5, 2($1)
        #ab hier compiler hwords, endianesstest
        lhu $10, h0
        lhu $11, h1
idle:
        j idle

.text
var: .word 0x80818283
h0: .half 0x4444
h1: .half 0x5555
```

Register	Inhalt	OK
1	0x2C	Scf
4	0x8081	Scf
5	0x8284	Scf
10	0x4444	Scf
11	0x5555	Scf

lh:

```
main:
        la $1, var #load address
        lh $4, var
        lh $5, 2($1)
        #ab hier compiler hwords, endianesstest
        lh $10, h0
        lh $11, h1
idle:
        j idle

.text
var: .word 0x8081
h0: .half 0x4444
h1: .half 0x8555
```

Register	Inhalt	OK
1	0x0000 0002C	Scf
4	0xFFFF 8081	Scf
5	0x4283	Scf
10	0x4444	Scf
11	0xFFFF 8555	Scf

lbu:

```
main:
        la $1, var #load address
        lbu $2, var
        addi $1, $1, 1
        lbu $3, 0($1)
        addi $1, $1, 1
        lbu $4, 0($1)
        lbu $5, 1($1)
        #ab hier compilerbytes, endianesstest
        lbu $10, b0
        lbu $11, b1
        lbu $12, b2
        lbu $13, b3
idle:
        j idle

.text
var: .word 0x80414283
b0: .byte 0x84
b1: .byte 0x45
b2: .byte 0x86
b3: .byte 0x47
```

Register	Inhalt	OK
1	0x4C / 0x4D / 0x4E	Scf
2	0x80	Scf
3	0x41	Scf
4	0x42	Scf
5	0x83	Scf
10	0x84	Scf
11	0x45	Scf
12	0x86	Scf
13	0x47	Scf

lb:

main:
```
    la $1, var #load address
    lb $2, var
    addi $1, $1, 1
    lb $3, 0($1)
    addi $1, $1, 1
    lb $4, 0($1)
    lb $5, 1($1)
    #ab hier compilerbytes, endianesstest
    lb $10, b0
    lb $11, b1
    lb $12, b2
    lb $13, b3
idle:
    j idle

.text
var: .word 0x80414283
b0: .byte 0x84
b1: .byte 0x45
b2: .byte 0x86
b3: .byte 0x47
```

Register	Inhalt	OK
1	0x4C / 0x4D / 0x4E	Scf
2	0xFFFF FF80	Scf
3	0x41	Scf
4	0x42	Scf
5	0xFFFF FF83	Scf
10	0xFFFF FF84	Scf
11	0x45	Scf
12	0xFFFF FF86	Scf
13	0x47	Scf

lwl:

```
main:
        la $1, var #load address
        lwl $2, 0($1)
        lwl $3, 1($1)
        lwl $4, 2($1)
        lwl $5, 3($1)
idle:
        j idle

.text
var: .word 0x494a4b4c
```

Register	Inhalt	OK
1	0x0000 0020	Scf
2	0x494a 4b4c	Scf
3	0x4a4b 4c00	Scf
4	0x4b4c 0000	Scf
5	0x4c00 0000	Scf

lwr:

main:
    la $1, var #load address
    lwr $2, 0($1)
    lwr $3, 1($1)
    lwr $4, 2($1)
    lwr $5, 3($1)
idle:
    j idle

.text
var: .word 0x494a4b4c

Register	Inhalt	OK
1	0x0000 0020	Scf
2	0x0000 0049	Scf
3	0x0000 494a	Scf
4	0x0049 4a4b	Scf
5	0x494a 4b4c	Scf

lwl_lwr:

```
main:
        la $1, var0 #load address
        lwl  $2, 2($1)
        lwr  $2, 5($1)
idle:
        j idle

.text
var0: .word 0x0000494a
var1: .word 0x4b4c0000
```

Register	Inhalt	OK
1	0x0000 0018	Scf
2	0x494A 4B4C	Scf

sw:

```
main:
        la $1, var #load address
        lw $2, var
        sw $2, 2000
        lw $3, 2000
        ori $10, $0, 1000
        lw $4, 1000($10)
        li $11, 0x00400000
        sw $2, 0($11)
        lw $6, 0($11)
idle:
        j idle

.text
var: .word 0x80818283
```

Register	Inhalt	OK
1	0x0000 0038	Scf
2	0x80818283	Scf
3	0x80818283	Scf
4	0x80818283	Scf
6	0x80818283	Scf
10	0x0000 03E8	Scf
11	0x0040 0000	Scf

sh:

```
main:
        lhu $1, h0
        lhu $2, h1
        ori $3, $0, 1000
        sh $1, 1000($3)
        sh $2, 1002($3)
        addi $10, $3, 1000
        lhu $5, 0($10)
        lhu $6, 2($10)
        li $11, 0x00400000
        sh $1, 0($11)
        sh $2, 2($11)
        lh $7, 0($11)
        lh $8, 2($11)
idle:
        j idle

.text
h0: .half 0x4444
h1: .half 0x8555
```

Register	Inhalt	OK
1	0x4444	Scf
2	0x8555	Scf
3	0x0000 03E8	Scf
5	0x4444	Scf
6	0x8555	Scf
7	0x4444	Scf
8	0xFFFF 8555	Scf
10	0x0000 07D0	Scf
11	0x0040 0000	Scf

sb:

```
main:
        lbu $1, b0
        lbu $2, b1
        lbu $3, b2
        lbu $4, b3
        ori $5, $0, 1000
        sb $1, 1000($5)
        sb $2, 1001($5)
        sb $3, 1002($5)
        sb $4, 1003($5)
        addi $6, $5, 1000
        lbu $10, 0($6)
        lbu $11, 1($6)
        lbu $12, 2($6)
        lbu $13, 3($6)
idle:
        j idle

.text
b0: .byte 0x44
b1: .byte 0x85
b2: .byte 0x46
b3: .byte 0x87
```

Register	Inhalt	OK
1	0x44	Scf
2	0x85	Scf
3	0x46	Scf
4	0x87	Scf
5	0x0000 03E8	Scf
6	0x0000 07D0	Scf
10	0x44	Scf
11	0x85	Scf
12	0x46	Scf
13	0x87	Scf

swl:

```
main:
        la $1, var #load address
        li $2, 0x69707172
        swl $2, 0($1)
        lw $5, 0($1)
        sw $0, 0($1) //nullen fuer naechste Runde
        swl $2, 1($1)
        lw $6, 0($1)
        sw $0, 0($1)
        swl $2, 2($1)
        lw $7, 0($1)
        sw $0, 0($1)
        swl $2, 3($1)
        lw $8, 0($1)
idle:
        j idle

.text
var: .word 0x00000000
```

Register	Inhalt	Offset
1	0x0000 0044	Scf
2	0x6970 7172	Scf
5	0x6970 7172	Scf
6	0x0069 7071	Scf
7	0x0000 6970	Scf
8	0x0000 0069	Scf

swr:

main:
```
    la $1, var #load address
    li $2, 0x69707172
    swr $2, 0($1)
    lw $5, 0($1)
    sw $0, 0($1) //nullen fuer naechste Runde
    swr $2, 1($1)
    lw $6, 0($1)
    sw $0, 0($1)
    swr $2, 2($1)
    lw $7, 0($1)
    sw $0, 0($1)
    swr $2, 3($1)
    lw $8, 0($1)
```
idle:
```
    j idle
```

.text
var: .word 0x00000000

Register	Inhalt	Offset
1	0x0000 0044	Scf
2	0x6970 7172	Scf
5	0x7200 0000	Scf
6	0x7172 0000	Scf
7	0x7071 7200	Scf
8	0x6970 7172	Scf

swl_lwl:

```
main:
        la $1, var0 #load address
        li $2, 0x69707172
        swl  $2, 2($1)
        swr  $2, 5($1)
        lw $4, 0($1)
        lw $5, 4($1)
idle:
        j idle

.text
var0: .word 0x00000000
var1: .word 0x00000000
```

Register	Inhalt	OK
1	0x0000 0028	Scf
2	0x6970 7172	Scf
4	0x0000 6970	Scf
5	0x7172 0000	Scf

Exceptions

break:

```
.section .text.reset
reset_vec:
        j main

.section .text.break
break_vec: (0x00000180)
        add $10, $2, $10
        addi $10, $0, 0xabcd
        rfe

.section .text
.globl main (0x00000400)
main:
        lui $1, 0xffff
        ori $1, $1, 0xfff4
        ori $2, $0, 0x42
        break
        addiu $3, $2, -1
        nop
        addu $4, $1, $2
idle:
        j idle
```

Register	Inhalt	OK
1	0xFFFF FFF4	Scf
2	0x42	Scf
3	0x41	Scf
4	0x36	Scf
10	0x42	Scf
11	0xFFFF ABCD	Scf
EPC	0x0000 0414	Scf

syscall:

```
.section .text.reset
reset_vec:
        j main

.section .text.sysc
break_sysc:
        add $10, $2, $10
        addi $11, $0, 0xabcd
        rfe

.section .text
.globl main
main:
        lui $1, 0xffff
        ori $1, $1, 0xfff4
        ori $2, $0, 0x42
        syscall 0
        addiu $3, $2, -1
        nop
        addu $4, $1, $2
idle:
        j idle
```

Register	Inhalt	OK
1	0xFFFF FFF4	Scf
2	0x42	Scf
3	0x41	Scf
4	0x36	Scf
10	0x42	Scf
11	0xFFFF ABCD	Scf
EPC	0x0000 0414	Scf

```
overflow:

reset_vec:
        j main

.section .text.ovw
ovw_vec:
        add $10, $2, $10
        rfe

.section .text
.globl main
main:
        lui $1, 0x8000
        ori $1, $1, 0x0000
        ori $2, $0, 0x42
        addi $11, $1, -1
        addu $3, $2, $2
        nop
        addu $4, $1, $2
idle:
        j idle
```

Register	Inhalt	OK
1	0x8000 0000	Scf
2	0x0000 0042	Scf
3	0x0000 0084	Scf
4	0x8000 0042	Scf
10	0x0000 0042	Scf
11	Unverändert	Scf
EPC	0x0000 0414	

```
undefined:

.section .text.reset
reset_vec:
        j main

.section .text.undef
break_undef:
        add $10, $2, $10
        addi $11, $0, 0xabcd
        rfe

.section .text
.globl main
main:
        lui $1, 0xffff
        ori $1, $1, 0xfff4
        ori $2, $0, 0x42
        .word 0xFFFFFFFF
        addiu $3, $2, -1
        nop
        addu $4, $1, $2
idle:
        j idle
```

Register	Inhalt	OK
1	0xFFFF FFF4	
2	0x42	
3	0x41	
4	0x36	
10	0x42	
11	0xFFFF ABCD	
EPC	0x0000 0414	

interrupt

cop2:

```
.section .text.reset
reset_vec:
        j main

.section .text.break
break_vec: (0x00000180)
        cop2 0x010000
        addi $10, $0, 0xabcd
        rfe

.section .text
.globl main
main: (0x00000400)
        lui $2, 0xffff
        ori $2, $2, 0xfff4
        ori $3, $0, 0x42
        break
        addiu $4, $3, -1
        nop
        addu $5, $2, $3
idle:
        j idle
```

Register	Inhalt	OK
1	0x0000 0410	Scf
2	0xFFFF FFF4	Scf
3	0x42	Scf
4	0x41	Scf
5	0x36	Scf
10	0xFFFF ABCD	Scf

HI LO Register

```
main:
        li $1, 0x80818283
        li $2, 0x70717273
        mthi $1
        mtlo $2
        mfhi $10
        mflo $11
idle:
        j idle
```

Register	Inhalt	OK
1	0x80818283	Scf
2	0x70717273	Scf
10	0x80818283	Scf
11	0x70717273	Scf

Punktrechnung

div:

```
.section .text.reset
reset_vec:
        li $sp, 0x8000
        j main

.section .text.div
div_vec:
        j divsim_sign

.section .text
.globl main
main:
        li $1, 7
        li $2, 4
        div $0, $1, $2
        mflo $10
        mfhi $11
        nop
idle:
        j idle
```

Register	Inhalt	OK
1	7	F
2	4	F
10	1	F
11	3	F
29	32768	F

divu:

```
.section .text.reset
reset_vec:
        li $sp, 0x8000
        j main

.section .text.divu
div_vecu:
        j divsim_unsign

.section .text
.globl main
main:
        li $1, 7
        li $2, 4
        divu $0, $1, $2
        mflo $10
        mfhi $11
        nop
idle:
        j idle
```

Register	Inhalt	OK
1	7	F
2	4	F
10	1	F
11	3	F
29	32768	F

mult:

```
.section .text.reset
reset_vec:
        li $sp, 0x8000
        j main

.section .text.mult
div_mult:
        j mulsim_signed

.section .text
.globl main
main:
        li $1, 7
        li $2, -4
        mult $1, $2
        mflo $10
        mfhi $11
        nop
idle:
        j idle
```

Register	Inhalt	OK
1	7	f
2	-4	F
10	-28	F
11	-1 (0x FFFF FFFF)	F
29	32768	f

multu:

```
.section .text.reset
reset_vec:
        li $sp, 0x8000
        j main

.section .text.mult
div_mult:
        j mulsim_signed

.section .text
.globl main
main:
        li $1, 7
        li $2, -4
        mult $1, $2
        mflo $10
        mfhi $11
        nop
idle:
        j idle
```

Register	Inhalt	OK
1	7	f
2	4	F
10	28	F
11	0	F
29	32768	f