<N26112437>_<劉兆軒> AIAS 2023 Lab 4 HW Submission

1. 請不要用這份template 交作業, 建立一個新的codimd 檔案, 然後copy & paste 這個 template 到你創建的檔案做修改。
2. 請修改你的學號與姓名在上面的 title, 以避免TA 修改作業時把檔案跟人弄錯了
3. 在Playlab 作業中心繳交作業時, 請用你創建的檔案鏈結繳交, 其他相關的資料與鏈結請依照Template 規定的格式記載於codimd 上。

記得在文件標題上修改你的 <學號> <姓名>

Gitlab code link

Please paste the link to your private Gitlab repository for this homework submission here.

- Gitlab link - https://playlab.computing.ncku.edu.tw:4001/kevin1217/lab4 (https://playlab.computing.ncku.edu.tw:4001/kevin1217/lab4)

HW4-1 - RISC-V M-Standard Extension

C code - MUL (範例)

請參考 Lab4-1和下方範例, **將新增的 code**放在下方並加上註解, 讓 TA明白你是如何完成的。

```
// C code you add & comment
// Please DON'T copy all your code, just copy the part you add

// Instructionuction : MUL
// Line 48
typedef enum {
UNIMPL = 0,
  MUL,
} instr_type;

//line 95
instr_type parse_instr(char* tok) {
  if ( streq(tok , "mul")) return MUL;
}

//line 522
switch( op ) {
    case UNIMPL: return 1;
    case MUL:
        if ( !o1 || !o2 || !o3 || o4 ) print_syntax_error( line,  "Invalid forma
        i->a1.reg = parse_reg(o1 , line);
        i->a2.reg = parse_reg(o2 , line);
        i->a3.reg = parse_reg(o3 , line);
        return 1;
}

//line 642
switch (i.op) {
    case MUL: rf[i.a1.reg] = rf[i.a2.reg] * rf[i.a3.reg]; break;
}

```

Simulation Result & Assembly Code

1. 請放上你用來驗證 instruction的 assembly code, 並加上預期結果的註解。
2. 使用 RV32 Emulator模擬, 驗證程式碼的正確性。
3. 用以驗證的 assembly code可以有不只一組, 也可以只有一組, 確保 function正確就好。

- Assembly code to test MUL function

```
main:
addi x28,x0 ,2      ## x28 = 2
addi x29,x0 ,3      ## x29 = 3
mul  x30,x28,x29    ## x30 = 2*3 = 6
hcf                 ## Terminate
```

- Simulation result

```
----------

Reached Halt and Catch Fire instruction!
inst:      4 pc:     12 src line: 7
x00:0x00000000 x01:0x00000000 x02:0x00000000 x03:0x00000000 x04:0x00000000 x05:0x00000000 x06:0x00000000 x07:0x00000000
x08:0x00000000 x09:0x00000000 x10:0x00000000 x11:0x00000000 x12:0x00000000 x13:0x00000000 x14:0x00000000 x15:0x00000000
x16:0x00000000 x17:0x00000000 x18:0x00000000 x19:0x00000000 x20:0x00000000 x21:0x00000000 x22:0x00000000 x23:0x00000000
x24:0x00000000 x25:0x00000000 x26:0x00000000 x27:0x00000000 x28:0x00000002 x29:0x00000003 x30:0x00000006 x31:0x00000000
Execution done!
user@aa62d5f4e749:/workspace/projects/lab4/HW4-1$
```

C code - MULHU (TODO)

> 請參考 Lab4-1和範例, **將新增的 code**放在下方並加上註解, 讓 TA明白你是如何完成的。

```c
// C code you add & comment
// Please DON'T copy all your code, just copy the part you add

// Instructionuction : MULHU
// Line 53
typedef enum {
UNIMPL = 0,
  MULHU,
} instr_type;

//line 102
instr_type parse_instr(char* tok) {
  if ( streq(tok , "mulhu")) return MULHU;
}

//line 541
switch( op ) {
    case UNIMPL: return 1;
    case MULHU:
        if ( !o1 || !o2 || !o3 || o4 ) print_syntax_error( line,  "Invalid forma
        i->a1.reg = parse_reg(o1 , line);
        i->a2.reg = parse_reg(o2 , line);
        i->a3.reg = parse_reg(o3 , line);
        return 1;
}

//line 801
switch (i.op) {
    case MULHU: rf[i.a1.reg] = (((uint64_t)rf[i.a2.reg] * (uint64_t)rf[i.a3.reg]
}
```

Simulation Result & Assembly Code

- Assembly code to test MULHU function

```
## RV32 Emulator Testing Assembly Code for MULHU function
main:
addi x28,x0 ,-1
addi x29,x0 ,-1
mulhu  x30,x28,x29
hcf
```

- Simulation result

```
Reached Halt and Catch Fire instruction!
inst:      4 pc:     12 src line: 6
x00:0x00000000 x01:0x00000000 x02:0x00000000 x03:0x00000000 x04:0x00000000 x05:0x00000000 x06:0x00000000 x07:0x00000000
x08:0x00000000 x09:0x00000000 x10:0x00000000 x11:0x00000000 x12:0x00000000 x13:0x00000000 x14:0x00000000 x15:0x00000000
x16:0x00000000 x17:0x00000000 x18:0x00000000 x19:0x00000000 x20:0x00000000 x21:0x00000000 x22:0x00000000 x23:0x00000000
x24:0x00000000 x25:0x00000000 x26:0x00000000 x27:0x00000000 x28:0xffffffff x29:0xffffffff x30:0xfffffffe x31:0x00000000
Execution done!
```

C code - REM (TODO)

```
// C code you add & comment
// Please DON'T copy all your code, just copy the part you add

// Instructionuction : REM
// Line 55
typedef enum {
UNIMPL = 0,
  REM,
} instr_type;

//line 103
instr_type parse_instr(char* tok) {
  if ( streq(tok , "rem")) return REM;
}

//line 549
switch( op ) {
    case UNIMPL: return 1;
    case REM:
        if ( !o1 || !o2 || !o3 || o4 ) print_syntax_error( line,  "Invalid forma
        i->a1.reg = parse_reg(o1 , line);
        i->a2.reg = parse_reg(o2 , line);
        i->a3.reg = parse_reg(o3 , line);
        return 1;
}

//line 802
switch (i.op) {
    case REM: rf[i.a1.reg] = (int32_t)rf[i.a2.reg] % (int32_t)rf[i.a3.reg]; brea
}

```

Simulation Result & Assembly Code

- Assembly code to test REM function

```
## RV32 Emulator Testing Assembly Code for REM function
main:
addi x28,x0 ,5
addi x29,x0 ,2
rem  x30,x28,x29
hcf
```

- Simulation result

```
Reached Halt and Catch Fire instruction!
inst:      4 pc:     12 src line: 7
x00:0x00000000 x01:0x00000000 x02:0x00000000 x03:0x00000000 x04:0x00000000 x05:0x00000000 x06:0x00000000 x07:0x00000000
x08:0x00000000 x09:0x00000000 x10:0x00000000 x11:0x00000000 x12:0x00000000 x13:0x00000000 x14:0x00000000 x15:0x00000000
x16:0x00000000 x17:0x00000000 x18:0x00000000 x19:0x00000000 x20:0x00000000 x21:0x00000000 x22:0x00000000 x23:0x00000000
x24:0x00000000 x25:0x00000000 x26:0x00000000 x27:0x00000000 x28:0x00000005 x29:0x00000002 x30:0x00000001 x31:0x00000000
Execution done!
```

C code - REMU (TODO)

> 請參考 Lab4-1和範例, **將新增的 code**放在下方並加上註解, 讓 TA明白你是如何完成的。

```
// C code you add & comment
// Please DON'T copy all your code, just copy the part you add

// Instructionuction : REMU
// Line 56
typedef enum {
UNIMPL = 0,
  REMU,
} instr_type;

//line 104
instr_type parse_instr(char* tok) {
  if ( streq(tok , "remu")) return REMU;
}

//line 555
switch( op ) {
    case UNIMPL: return 1;
    case REMU:
        if ( !o1 || !o2 || !o3 || o4 ) print_syntax_error( line,  "Invalid forma
        i->a1.reg = parse_reg(o1 , line);
        i->a2.reg = parse_reg(o2 , line);
        i->a3.reg = parse_reg(o3 , line);
        return 1;
}

//line 803
switch (i.op) {
    case REMU: rf[i.a1.reg] = rf[i.a2.reg] % rf[i.a3.reg]; break;
}

```

Simulation Result & Assembly Code

- Assembly code to test REMU function

```
## RV32 Emulator Testing Assembly Code for REMU function
main:
addi x28,x0 ,-1
addi x29,x0 ,2
remu  x30,x28,x29
hcf
```

- Simulation result

```
Reached Halt and Catch Fire instruction!
inst:      4 pc:     12 src line: 7
x00:0x00000000 x01:0x00000000 x02:0x00000000 x03:0x00000000 x04:0x00000000 x05:0x00000000 x06:0x00000000 x07:0x00000000
x08:0x00000000 x09:0x00000000 x10:0x00000000 x11:0x00000000 x12:0x00000000 x13:0x00000000 x14:0x00000000 x15:0x00000000
x16:0x00000000 x17:0x00000000 x18:0x00000000 x19:0x00000000 x20:0x00000000 x21:0x00000000 x22:0x00000000 x23:0x00000000
x24:0x00000000 x25:0x00000000 x26:0x00000000 x27:0x00000000 x28:0xffffffff x29:0x00000002 x30:0x00000001 x31:0x00000000
Execution done!
```

HW4-2 - RISC-V Bit Manipulation Extension

Gitlab code link (Your own branch)

Please paste the link to your branch of group Gitlab repository for this homework submission here.

- Gitlab link of your branch - https://playlab.computing.ncku.edu.tw:4001/s8942352/lab4-group/tree/kevin1217 (https://playlab.computing.ncku.edu.tw:4001/s8942352/lab4-group/tree/kevin1217)

Please paste the link to your group project repository for this homework submission here.

- Gitlab link of your group project repo - https://playlab.computing.ncku.edu.tw:4001/s8942352/lab4-group/tree/master (https://playlab.computing.ncku.edu.tw:4001/s8942352/lab4-group/tree/master)

Please add a project.md (http://project.md) file in your group repo. In project.bd (http://project.bd) please write down the list of your member & distributions of each member.

C Code - SEXTB

> 請參考 Lab4-1和範例, **將新增的 code**放在下方並加上註解, 讓 TA明白你是如何完成的, 如果超過一個指令請依照 HW4-1的方式新增作業說明。

```
// C code you add & comment
// Please DON'T copy all your code, just copy the part you add

// Instructionuction : SEXTB
// Line 69
typedef enum {
UNIMPL = 0,
  SEXTB,
} instr_type;

//line 149
instr_type parse_instr(char* tok) {
  if ( streq(tok , "sext.b")) return SEXTB;
}

//line 727
switch( op ) {
    case UNIMPL: return 1;
    case SEXTB:
            if ( !o1 || !o2 || o3 || o4 ) print_syntax_error( line,  "Invalid fo
                i->a1.reg = parse_reg(o1 , line);
                i->a2.reg = parse_reg(o2 , line);
            return 1;


//line 1134
switch (i.op) {
    case SEXTB:
                //sll 左移補0
                rf[i.a1.reg] = rf[i.a2.reg] << 24;
                //sra算術右移有號延伸
                rf[i.a1.reg] = (*(int32_t*)&rf[i.a1.reg]) >> 24;
                break;

}
```

Simulation Result & Assembly Code

- Assembly code to test instruction you picked

```
main:
li x28,895
sext.b  x30,x28
hcf
```

- Simulation result

```
Reached Halt and Catch Fire instruction!
inst:      4 pc:     12 src line: 4
x00:0x00000000 x01:0x00000000 x02:0x00000000 x03:0x00000000 x04:0x00000000 x05:0x00000000 x06:0x00000000 x07:0x00000000
x08:0x00000000 x09:0x00000000 x10:0x00000000 x11:0x00000000 x12:0x00000000 x13:0x00000000 x14:0x00000000 x15:0x00000000
x16:0x00000000 x17:0x00000000 x18:0x00000000 x19:0x00000000 x20:0x00000000 x21:0x00000000 x22:0x00000000 x23:0x00000000
x24:0x00000000 x25:0x00000000 x26:0x00000000 x27:0x00000000 x28:0x0000037f x29:0x00000000 x30:0x0000007f x31:0x00000000
Execution done!
```

C Code - SEXTH

> 請參考 Lab4-1和範例, **將新增的 code**放在下方並加上註解, 讓 TA明白你是如何完成的, 如果超過一個指令請依照 HW4-1的方式新增作業說明。

```c
// C code you add & comment
// Please DON'T copy all your code, just copy the part you add

// Instructionuction : SEXTH
// Line 70
typedef enum {
UNIMPL = 0,
  SEXTH,
} instr_type;

//line 150
instr_type parse_instr(char* tok) {
  if ( streq(tok , "sext.h")) return SEXTH;
}

//line 732
switch( op ) {
    case UNIMPL: return 1;
    case SEXTH:
            if ( !o1 || !o2 || o3 || o4 ) print_syntax_error( line,  "Invalid fo
                i->a1.reg = parse_reg(o1 , line);
                i->a2.reg = parse_reg(o2 , line);
            return 1;


//line 1140
switch (i.op) {
    case SEXTH:
            //sll 左移補0
            rf[i.a1.reg] = rf[i.a2.reg] << 16;
            //sra(arithmetic)算術右移有號延伸
            rf[i.a1.reg] = (*(int32_t*)&rf[i.a1.reg]) >> 16;
            break;

}
```

Simulation Result & Assembly Code

- Assembly code to test instruction you picked

```
main:
li x28,32768
sext.h x30,x28
hcf
```

- Simulation result

```
Reached Halt and Catch Fire instruction!
inst:      4 pc:     12 src line: 4
x00:0x00000000 x01:0x00000000 x02:0x00000000 x03:0x00000000 x04:0x00000000 x05:0x00000000 x06:0x00000000 x07:0x00000000
x08:0x00000000 x09:0x00000000 x10:0x00000000 x11:0x00000000 x12:0x00000000 x13:0x00000000 x14:0x00000000 x15:0x00000000
x16:0x00000000 x17:0x00000000 x18:0x00000000 x19:0x00000000 x20:0x00000000 x21:0x00000000 x22:0x00000000 x23:0x00000000
x24:0x00000000 x25:0x00000000 x26:0x00000000 x27:0x00000000 x28:0x00008000 x29:0x00000000 x30:0xffff8000 x31:0x00000000
Execution done!
```

C Code - ZEXTH

> 請參考 Lab4-1和範例, **將新增的 code**放在下方並加上註解, 讓 TA明白你是如何完成的, 如果超過一個指令請依照 HW4-1的方式新增作業說明。

```
// C code you add & comment
// Please DON'T copy all your code, just copy the part you add

// Instructionuction : ZEXTH
// Line 71
typedef enum {
UNIMPL = 0,
  ZEXTH,
} instr_type;

//line 151
instr_type parse_instr(char* tok) {
    if ( streq(tok , "zext.h")) return ZEXTH;
}

//line 737
switch( op ) {
    case UNIMPL: return 1;
    case ZEXTH:
            if ( !o1 || !o2 || o3 || o4 ) print_syntax_error( line,  "Invalid fo
                i->a1.reg = parse_reg(o1 , line);
                i->a2.reg = parse_reg(o2 , line);
            return 1;


//line 1146
switch (i.op) {
    case ZEXTH:
            //sll 左移補0
            rf[i.a1.reg] = rf[i.a2.reg] << 16;
            //srl算術右移無號延伸
            rf[i.a1.reg] = rf[i.a1.reg] >> 16;


}
```

Simulation Result & Assembly Code

- Assembly code to test instruction you picked

```
main:
li x28,120000
zext.h x30,x28
hcf
```

- Simulation result

```
Reached Halt and Catch Fire instruction!
inst:      4 pc:     12 src line: 4
x00:0x00000000 x01:0x00000000 x02:0x00000000 x03:0x00000000 x04:0x00000000 x05:0x00000000 x06:0x00000000 x07:0x00000000
x08:0x00000000 x09:0x00000000 x10:0x00000000 x11:0x00000000 x12:0x00000000 x13:0x00000000 x14:0x00000000 x15:0x00000000
x16:0x00000000 x17:0x00000000 x18:0x00000000 x19:0x00000000 x20:0x00000000 x21:0x00000000 x22:0x00000000 x23:0x00000000
x24:0x00000000 x25:0x00000000 x26:0x00000000 x27:0x00000000 x28:0x0001d4c0 x29:0x00000000 x30:0x0000d4c0 x31:0x00000000
Execution done!
```

C Code - CLMUL

> 請參考 Lab4-1和範例, **將新增的 code**放在下方並加上註解, 讓 TA明白你是如何完成的, 如果超過一個指令請依照 HW4-1的方式新增作業說明。

```
// C code you add & comment
// Please DON'T copy all your code, just copy the part you add

// Instructionuction : CLMUL
// Line 72
typedef enum {
UNIMPL = 0,
  CLMUL,
} instr_type;

//line 152
instr_type parse_instr(char* tok) {
  if ( streq(tok , "clmul")) return CLMUL;
}

//line 742
switch( op ) {
    case UNIMPL: return 1;
    case CLMUL:
        if ( !o1 || !o2 || !o3 || o4 ) print_syntax_error( line,  "Invalid forma
            i->a1.reg = parse_reg(o1 , line);
            i->a2.reg = parse_reg(o2 , line);
            i->a3.reg = parse_reg(o3 , line);
        return 1;


//line 1152
switch (i.op) {
    case CLMUL:
            for(int j=0;j<=32;j++)
            {
                if  (( rf[i.a3.reg] >> j & 1)!=0)
                {
                    rf[i.a1.reg] = rf[i.a1.reg] ^ (rf[i.a2.reg] << j);
                }
            }
            break;

```

Simulation Result & Assembly Code

- Assembly code to test instruction you picked

```
main:
li x28,3
li x29,3
CLMUL x30,x28,x29
hcf
```

- Simulation result

```
Reached Halt and Catch Fire instruction!
inst:      6 pc:     20 src line: 5
x00:0x00000000 x01:0x00000000 x02:0x00000000 x03:0x00000000 x04:0x00000000 x05:0x00000000 x06:0x00000000 x07:0x00000000
x08:0x00000000 x09:0x00000000 x10:0x00000000 x11:0x00000000 x12:0x00000000 x13:0x00000000 x14:0x00000000 x15:0x00000000
x16:0x00000000 x17:0x00000000 x18:0x00000000 x19:0x00000000 x20:0x00000000 x21:0x00000000 x22:0x00000000 x23:0x00000000
x24:0x00000000 x25:0x00000000 x26:0x00000000 x27:0x00000000 x28:0x00000003 x29:0x00000003 x30:0x00000005 x31:0x00000000
Execution done!
```

C Code - CLMULH

> 請參考 Lab4-1和範例, **將新增的 code**放在下方並加上註解, 讓 TA明白你是如何完成的, 如果超過一個指令請依照 HW4-1的方式新增作業說明。

```
// C code you add & comment
// Please DON'T copy all your code, just copy the part you add

// Instructionuction : CLMULH
// Line 73
typedef enum {
UNIMPL = 0,
  CLMULH,
} instr_type;

//line 153
instr_type parse_instr(char* tok) {
    if ( streq(tok , "clmulh")) return CLMULH;
}

//line 748
switch( op ) {
    case UNIMPL: return 1;
    case CLMULH:
        if ( !o1 || !o2 || !o3 || o4 ) print_syntax_error( line,  "Invalid forma
            i->a1.reg = parse_reg(o1 , line);
            i->a2.reg = parse_reg(o2 , line);
            i->a3.reg = parse_reg(o3 , line);
        return 1;



//line 1162
switch (i.op) {
    case CLMULH:
            for(int j=1;j<=32;j++)
            {
                if  ( rf[i.a3.reg] >> j & 1)
                {
                    rf[i.a1.reg] = rf[i.a1.reg] ^ (rf[i.a2.reg] >> (32-j));
                }
            }
            break;

}
```

Simulation Result & Assembly Code

- Assembly code to test instruction you picked

```
main:
li x28,-1
li x29,2
CLMULH x30,x28,x29
hcf
```

- Simulation result

```
Reached Halt and Catch Fire instruction!
inst:      6 pc:     20 src line: 5
x00:0x00000000 x01:0x00000000 x02:0x00000000 x03:0x00000000 x04:0x00000000 x05:0x00000000 x06:0x00000000 x07:0x00000000
x08:0x00000000 x09:0x00000000 x10:0x00000000 x11:0x00000000 x12:0x00000000 x13:0x00000000 x14:0x00000000 x15:0x00000000
x16:0x00000000 x17:0x00000000 x18:0x00000000 x19:0x00000000 x20:0x00000000 x21:0x00000000 x22:0x00000000 x23:0x00000000
x24:0x00000000 x25:0x00000000 x26:0x00000000 x27:0x00000000 x28:0xffffffff x29:0x00000002 x30:0x00000001 x31:0x00000000
Execution done!
```

C Code - CLMULR

> 請參考 Lab4-1和範例, **將新增的 code**放在下方並加上註解, 讓 TA明白你是如何完成的, 如果超過一個指令請依照 HW4-1的方式新增作業說明。

```
// C code you add & comment
// Please DON'T copy all your code, just copy the part you add

// Instructionuction : CLMULR
// Line 74
typedef enum {
UNIMPL = 0,
  CLMULR,
} instr_type;

//line 154
instr_type parse_instr(char* tok) {
  if ( streq(tok , "clmulr")) return CLMULR;
}

//line 754
switch( op ) {
    case UNIMPL: return 1;
    case CLMULR:
        if ( !o1 || !o2 || !o3 || o4 ) print_syntax_error( line,  "Invalid forma
            i->a1.reg = parse_reg(o1 , line);
            i->a2.reg = parse_reg(o2 , line);
            i->a3.reg = parse_reg(o3 , line);
        return 1;



//line 1172
switch (i.op) {
    case CLMULR:
            for(int j=0;j<32;j++)
            {
                if  ( rf[i.a3.reg] >> j & 1)
                {
                    rf[i.a1.reg] = rf[i.a1.reg] ^ (rf[i.a2.reg] >> (32-j-1));
                }
            }
            break;


}
```

Simulation Result & Assembly Code

- Assembly code to test instruction you picked

```
main:
li x28,-1
li x29,3
CLMULR x30,x28,x29
hcf
```

- Simulation result

```
Reached Halt and Catch Fire instruction!
inst:      6 pc:     20 src line: 5
x00:0x00000000 x01:0x00000000 x02:0x00000000 x03:0x00000000 x04:0x00000000 x05:0x00000000 x06:0x00000000 x07:0x00000000
x08:0x00000000 x09:0x00000000 x10:0x00000000 x11:0x00000000 x12:0x00000000 x13:0x00000000 x14:0x00000000 x15:0x00000000
x16:0x00000000 x17:0x00000000 x18:0x00000000 x19:0x00000000 x20:0x00000000 x21:0x00000000 x22:0x00000000 x23:0x00000000
x24:0x00000000 x25:0x00000000 x26:0x00000000 x27:0x00000000 x28:0xffffffff x29:0x00000003 x30:0x00000002 x31:0x00000000
Execution done!
```

Bonus

> Bonus請依照 lab4 document中的 bonus template進行繳交。