마이크로프로세서

- Conditional Branch (JZ) -

Daejin Park

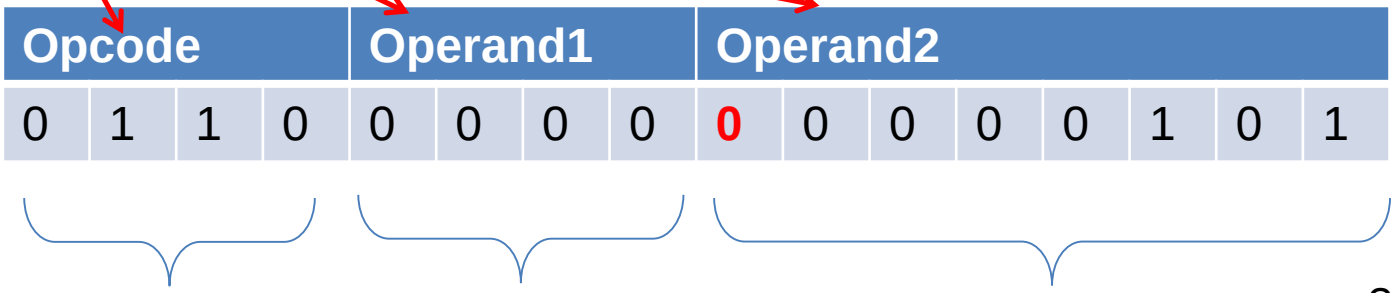
School of Electronics Engineering, KNU, KOREA

2019.05.01

Conditional Branch: JZ

- JZ R0, 5

0~127 (양수 128개)
-1~-128 (음수 128개)



Jump

If Rn is zero

-128~127 값 표현 가능
(데이터)

01111111	127
~	
00000010	2
00000001	1
00000000	0
11111111	-1
11111110	-2
~	
10000001	-127
10000000	-128

Decode 일부수정

Opcode: 6

CDecode.h

```
enum { MOV0=0, MOV1, MOV2, MOV3, ADD, SUB, JZ, MUL };
```

CDecode.cpp

```
void
CT1DecodeDirectFetch::show_instruction() {
    if(m_instruction.OPCODE == MOV3)  {
        cout << "MOV3 " << "R" << m_instruction.OP1 << ", #" << m_instruction.OP2 << endl;
    } else if(m_instruction.OPCODE == ADD )  {
        unsigned int op2 = (m_instruction.OP2 >> 4) & 0xF;
        cout << "ADD  " << "R" << m_instruction.OP1 << ", R" << op2 << endl;
    } else if(m_instruction.OPCODE == SUB )  {
        unsigned int op2 = (m_instruction.OP2 >> 4) & 0xF;
        cout << "SUB  " << "R" << m_instruction.OP1 << ", R" << op2 << endl;
    } else if(m_instruction.OPCODE == MOV0)  {
        unsigned int op2 = m_instruction.OP2 & 0xFF;
        cout << "MOV0 " << "R" << m_instruction.OP1 << ", [" << op2 << "]" << endl;
    } else if(m_instruction.OPCODE == MOV1)  {
        unsigned int op2 = m_instruction.OP2 & 0xFF;
        cout << "MOV1 " << "[" << op2 << "], R" << m_instruction.OP1 << endl;
    } else if(m_instruction.OPCODE == MUL )  {
        unsigned int op2 = (m_instruction.OP2 >> 4) & 0xF;
        cout << "MUL  " << "R" << m_instruction.OP1 << ", R" << op2 << endl;
    } else if(m_instruction.OPCODE == JZ ) {
        cout << "JZ   " << "R" << m_instruction.OP1 << ", " << m_instruction.OP2 << endl;
    }

}
```

Register File에 PC 추가

- PC (Program Counter) Register

현재 코드 메모리에 접근중인 주소

```
 2 #include <iostream>
 3
 4 #pragma once
 5
 6 using namespace std;
 7
 8 class CRegister {
 9 public:
10     CRegister() { }
11     virtual ~CRegister() { }
12
13 };
14
15 class C16RegisterFile : public CRegister {
16 public:
17     C16RegisterFile() : m_PC(0) {}
18     virtual ~C16RegisterFile() {}
19
20     void write_on_reg(unsigned int index, int data) { m_regs[index] = data; }
21     int read_from_reg(unsigned int index)           { return m_regs[index]; }
22
23     int get_PC() { return m_PC; }
24     void set_PC(int pc) { m_PC = pc; }
25
26     void show_regs();
27 private:
28     int m_regs[16];
29
30     int m_PC;
31 };
32
```

Execute 수정

CExecute.cpp

현재 명령어 실행 후
자동적으로 PC++

```
bool CT1ExecuteTinyUnit::do_execute() {

    // ex. MOV3 R0, #3
    if(m_decode_unit.get_opcode() == MOV3) {
        unsigned int reg_index = m_decode_unit.get_op1();
                 int      data = m_decode_unit.get_op2();

        m_regs.write_on_reg(reg_index, data);

        m_regs.set_PC(m_regs.get_PC()+1);

        return true;

      // ex. ADD R0, R1  --> R0 = R0 + R1
    } else if( m_decode_unit.get_opcode() == ADD) {
        unsigned int reg_n = m_decode_unit.get_op1();
        unsigned int reg_m = (m_decode_unit.get_op2() >> 4) & 0xF;

        int Rn = m_regs.read_from_reg(reg_n);
        int Rm = m_regs.read_from_reg(reg_m);

        Rn = Rn + Rm;
        m_regs.write_on_reg(reg_n, Rn);

        m_regs.set_PC(m_regs.get_PC()+1);

        return true;
```

Execute 수정

CExecute.cpp

현재 명령어 실행 후
자동적으로 PC++

```
  // ex. SUB R0, R1  --> R0 = R0 - R1
} else if( m_decode_unit.get_opcode() == SUB) {
    unsigned int reg_n = m_decode_unit.get_op1();
    unsigned int reg_m = (m_decode_unit.get_op2() >> 4) & 0xF;

    int Rn = m_regs.read_from_reg(reg_n);
    int Rm = m_regs.read_from_reg(reg_m);

    Rn = Rn - Rm;
    m_regs.write_on_reg(reg_n, Rn);

    m_regs.set_PC(m_regs.get_PC()+1);

    return true;


  // ex. MOV0 R1, [3] : R1 <- M[3]
} else if( m_decode_unit.get_opcode() == MOV0 ) {
    unsigned int reg_n    = m_decode_unit.get_op1();
    unsigned int mem_addr = m_decode_unit.get_op2() & 0xFF;


    int memory_data = m_mems.read_from_memory(mem_addr);

    m_regs.write_on_reg(reg_n, memory_data);

    m_regs.set_PC(m_regs.get_PC()+1);

    return true;
```

Execute 수정

CExecute.cpp

현재 명령어 실행 후
자동적으로 PC++

```
  // ex. MOV1 [3], R1 : M[3] <- R1
} else if( m_decode_unit.get_opcode() == MOV1 ) {
    unsigned int reg_n    = m_decode_unit.get_op1();
    unsigned int mem_addr = m_decode_unit.get_op2() & 0xFF;

    int Rn = m_regs.read_from_reg(reg_n);

    m_mems.write_on_memory(mem_addr, Rn);

    m_regs.set_PC(m_regs.get_PC()+1);

    return true;

  // ex. MUL R0, R1  --> R0 = R0 * R1
} else if( m_decode_unit.get_opcode() == MUL) {
    unsigned int reg_n = m_decode_unit.get_op1();
    unsigned int reg_m = (m_decode_unit.get_op2() >> 4) & 0xF;

    int Rn = m_regs.read_from_reg(reg_n);
    int Rm = m_regs.read_from_reg(reg_m);

    Rn = Rn * Rm;
    m_regs.write_on_reg(reg_n, Rn);

    m_regs.set_PC(m_regs.get_PC()+1);

    return true;
```

Execute 수정

CExecute.cpp

```
    // ex. JZ R0, #3 (R0 == 0, then PC+1+3)
    } else if(m_decode_unit.get_opcode() == JZ) {

        unsigned int reg_n  = m_decode_unit.get_op1();
                 int offset = m_decode_unit.get_op2();

        int Rn = m_regs.read_from_reg(reg_n);
        m_regs.set_PC(m_regs.get_PC()+1);

        if( Rn == 0 ) {
            int pc = m_regs.get_PC();
            m_regs.set_PC(pc+offset);
        }

        return true;

    } else {
        cout << "Not executable instruction, not yet implemented, sorry !!. " << endl;
        return false;
    }
}
```

현재 명령어 실행 후
자동적으로 PC++

Rn값이 Zero이면,
PC←PC+offset

테스트 코드

file.bin

```
MOV3 R0, #1
MOV3 R1, #1
MOV3 R2, #2
MOV3 R3, #0
ADD  R3, R2
ADD  R3, R2
SUB  R3, R1
MOV1 [1], R3
MOV0 R7, [1]
MOV1 [4], R2
MOV0 R8, [4]
MOV3 R14, #7
MOV3 R15, #1
ADD  R15, R14
MOV3 R13, #-1
MOV3 R12, #-2
MOV1 [255], R12
MOV1 [254], R13
MOV0 R11, [254]
```

```
0011000000000001
0011000100000001
0011001000000010
0011001100000000
0100001100100000
0100001100100000
0101001100010000
0001001100000001
0000011100000001
0001001000000100
0000100000000100
0011111000000111
0011111100000001
0100111111100000
0011110111111111
0011110011111110
0001110011111111
0001110111111110
0000101111111110
```

```
 1 MOV3 R0, #0
 2 JZ   R0, 5
 3 MOV3 R1, #1
 4 MOV3 R2, #2
 5 MOV3 R3, #0
 6 ADD  R3, R2
 7 ADD  R3, R2
 8 SUB  R3, R1
 9 MOV1 [1], R3
10 MOV0 R7, [1]
11 MOV1 [4], R2
12 MOV0 R8, [4]
13 MOV3 R14, #7
14 MOV3 R15, #1
15 ADD  R15, R14
16 MOV3 R13, #-1
17 MOV3 R12, #-2
18 MOV1 [255], R12
19 MOV1 [254], R13
20 MOV0 R11, [254]
```

PC

+5

PC

```
 1 0011000000000000
 2 0110000000000101
 3 0011000100000001
 4 0011001000000010
 5 0011001100000000
 6 0100001100100000
 7 0100001100100000
 8 0101001100010000
 9 0001001100000001
10 0000011100000001
11 0001001000000100
12 0000100000000100
13 0011111000000111
14 0011111100000001
15 0100111111100000
16 0011110111111111
17 0011110011111110
18 0001110011111111
19 0001110111111110
20 0000101111111110
```

TPU 실행

main.cpp

```
28      CT1DecodeDirectFetch decode(code_memory);
29      C16RegisterFile      regs;
30      CSRAM_256W           mems;
31
32      CT1ExecuteTinyUnit   execute(decode, regs, mems);
33
34      int size = atoi(argv[2]);
35  // for(int i=0; i<atoi(argv[2]); i++) {
36      while( regs.get_PC() < size ) {
37          decode.do_fetch_from(regs.get_PC());
38        //decode.do_fetch_from(i);
39          decode.do_decode();
40          decode.show_instruction();
41
42          execute.do_execute();
43      }
44
45      cout << "After executing instruction ..." << endl;
46      regs.show_regs();
47
48      //mems.show_mems(0, 9);
49      //mems.show_mems(250, 255);
50
51      mems.show_mems(0, 15);
52  }
53
```

실행결과

jump

```
MOV3  R0, #1
JZ    R0, 5
MOV3  R1, #1
MOV3  R2, #2
MOV3  R3, #0
ADD   R3, R2
ADD   R3, R2
SUB   R3, R1
MOV1  [1], R3
MOV0  R7, [1]
MOV1  [4], R2
MOV0  R8, [4]
MOV3  R14, #7
MOV3  R15, #1
ADD   R15, R14
MOV3  R13, #-1
MOV3  R12, #-2
MOV1  [255], R12
MOV1  [254], R13
MOV0  R11, [254]
```

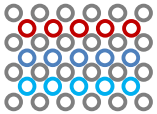
```
MOV3  R0, #0
JZ    R0, 5
SUB   R3, R1
MOV1  [1], R3
MOV0  R7, [1]
MOV1  [4], R2
MOV0  R8, [4]
MOV3  R14, #7
MOV3  R15, #1
ADD   R15, R14
MOV3  R13, #-1
MOV3  R12, #-2
MOV1  [255], R12
MOV1  [254], R13
MOV0  R11, [254]
```

Makefile

```
1 all:
2     g++ -o tpu CCode.cpp CDecode.cpp CExecute.cpp CRegist
  er.cpp CMemory.cpp main.cpp
3
4 #    ./tpu file.bin 19
5 #    ./tpu matrix.bin 40
6      ./tpu file2.bin 20
7
```

Q & A

Thank you for your attention



Architecture and Compiler
for Embedded Systems Lab.

School of Electronics Engineering, KNU

ACES Lab (boltanut@knu.ac.kr)