마이크로프로세서

– 메모리 추가 (MOV0, MOV1 명령어) –

Daejin Park

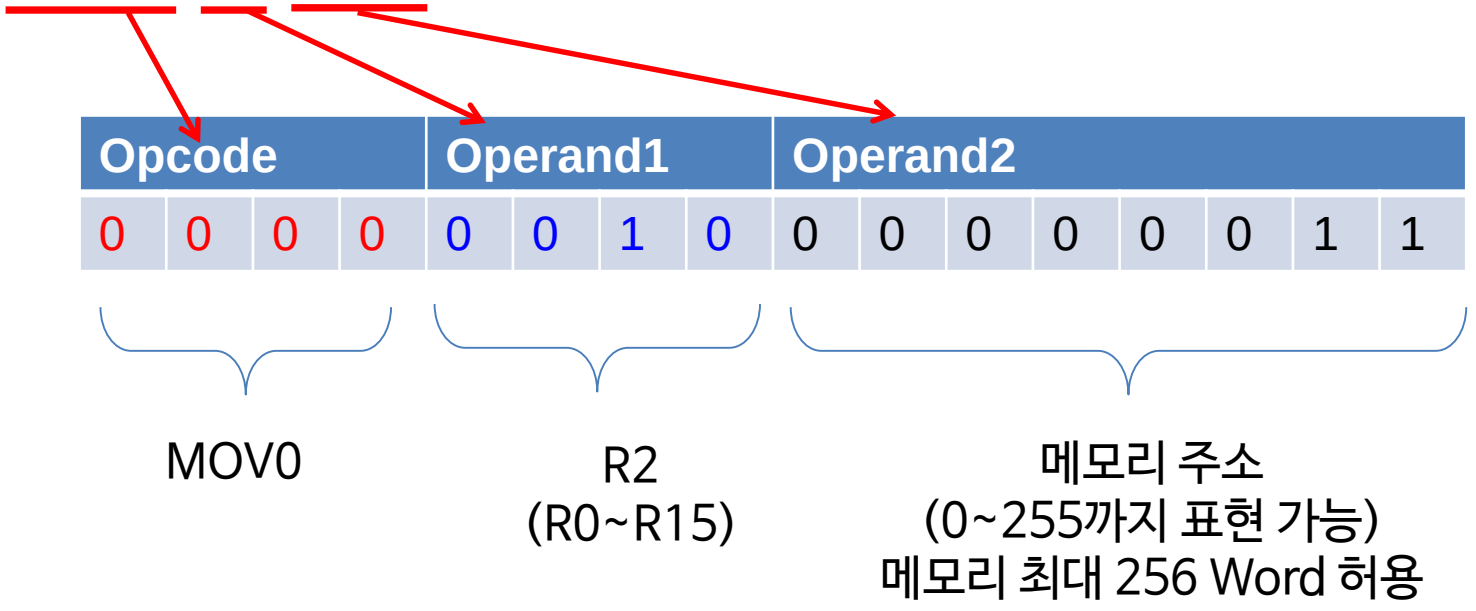
School of Electronics Engineering, KNU, KOREA

2019.04.10

Memory Read Instruction

- MOV0 Rn, [addr]
  - Rn ← M[addr]

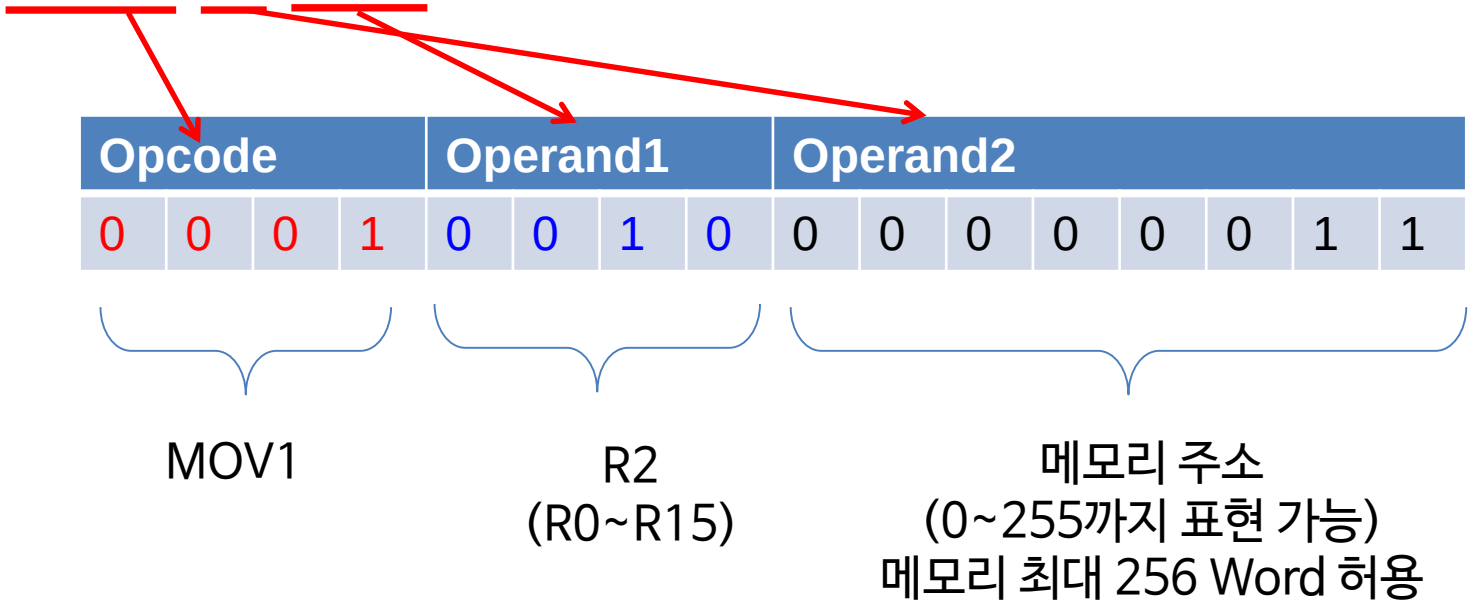
- Ex. MOV0 R2, [3]



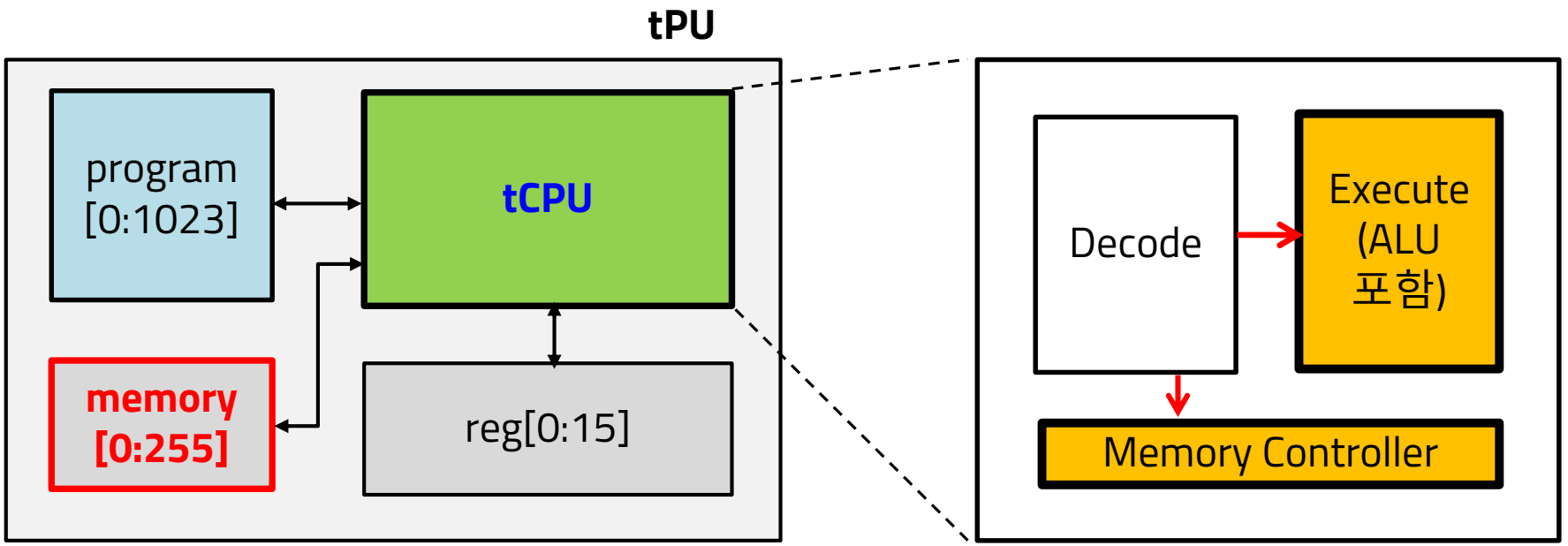
Memory Write Instruction

- MOV1 [addr], Rn
  - M[addr] ← Rn

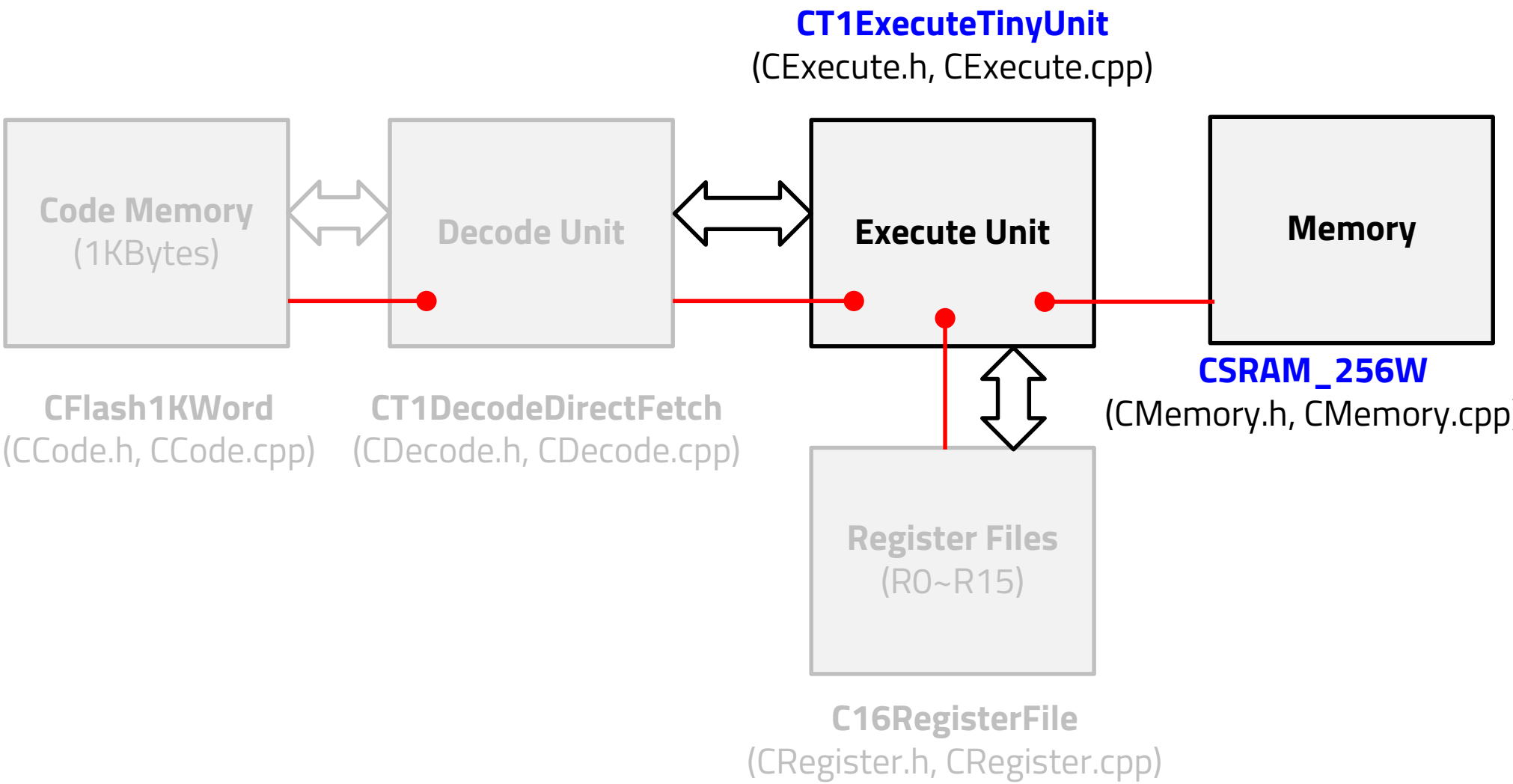
- Ex. MOV1 [3], R2



Controlling Execute Unit and Memory



Overall Architecture



Memory Unit

CMemory.h

```
#include <iostream>

#pragma once

using namespace std;

class CMemory {
public:
    CMemory() {}
    virtual ~CMemory() {}

};

class CSRAM_256W : public CMemory {
public:
    CSRAM_256W() { }
    virtual ~CSRAM_256W() { }

    void write_on_memory(unsigned int index, int data) { m_mems[index] = data; }
    int  read_from_memory(unsigned int index)          { return m_mems[index]; }

    void show_mems(unsigned char start_addr, unsigned char end_addr);

private:
    int m_mems[256];

};
```

Lab

Execution Unit에 Memory 추가

CExecute.h

```
#include <iostream>
#include "CDecode.h"
#include "CRegister.h"
#include "CMemory.h"

...

class CT1ExecuteTinyUnit: public CExecute {
public:
    CT1ExecuteTinyUnit(CT1DecodeDirectFetch& decode,
                       C16RegisterFile& regs,
                       CSRAM_256W& mems)
        : m_decode_unit(decode), m_regs(regs), m_mems(mems) { }
    virtual ~CT1ExecuteTinyUnit() { }

    bool do_execute();
private:

    CT1DecodeDirectFetch& m_decode_unit;
    C16RegisterFile&      m_regs;
    CSRAM_256W&           m_mems;
};
```

Execution Unit 에 MOV0, MOV1 기능 추가

CExecute.cpp

```
  // 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();


    int memory_data = m_mems.read_from_memory(mem_addr);

   m_regs.write_on_reg(reg_n, memory_data);

    return true;

  // ex. MOV1 [3], R1 : M[3] <- R1
} else if( m_decode_unit.get_opcode() == MOV1 ) {
    Lab

    return true;

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

Decode 일부수정

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;
    cout << "ADD  " << "R" << m_instruction.OP1 << ", R" << op2 << endl;

  } else if(m_instruction.OPCODE == SUB )  {
    unsigned int op2 = m_instruction.OP2 >> 4;
    cout << "SUB  " << "R" << m_instruction.OP1 << ", R" << op2 << endl;
```

추가

```
  } else if(m_instruction.OPCODE == MOV0)  {
    cout << "MOV0 R" << m_instruction.OP1 << ", [" << m_instruction.OP2 << "]" << endl;

  } else if(m_instruction.OPCODE == MOV1)  {
    cout << "MOV1 [" << m_instruction.OP2 << "], R" << m_instruction.OP1 << endl;
  }

}
```

Processor Top

main.cpp

```
    ...
    CT1DecodeDirectFetch decode(code_memory);
    C16RegisterFile      regs;
    CSRAM_256W           mems;

    CT1ExecuteTinyUnit   execute(decode, regs, mems);

    for(int i=0; i<atoi(argv[2]); i++) {
        decode.do_fetch_from(i);
        decode.do_decode();
        decode.show_instruction();

        execute.do_execute();
    }

    cout << "After executing instruction ..." << endl;
    regs.show_regs();
    mems.show_mems(0, 9);
}
```

함수 구현 못할 시 주석처리 하기 바람

기계코드 변환

- 실습

```
int main() {

    int m[10];

    int a = 1;
    int b = 1;
    int c = 2;
    int d = 0;
    int g, h;

    d = d + c; // 2
    d = d + c; // 4
    d = d – b; //3

    m[1] = d;
    g = m[1]; // g: 3

    m[4] = c;
    h = m[4]; // h: 2

}
```

compilation

```
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]
```

assembling

file.bin 수정 (직접)

opcode	op1	op2
00110000		00000001
00110001		00000001
00110010		00000010
00110011		00000000
01000011		00100000
01000011		00100000
01010011		00010000
00010011		00000001
00000111		00000001
00010010		00000100
00001000		00000100

Makefile

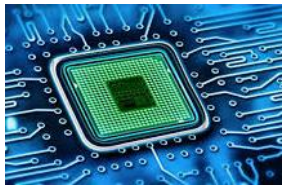
```
all:
   g++ -o tpu CCode.cpp CDecode.cpp CExecute.cpp CRegister.cpp CMemory.cpp main.cpp


  ./tpu file.bin 11
```

추가

프로세서 (HW)

펌웨어 (SW)

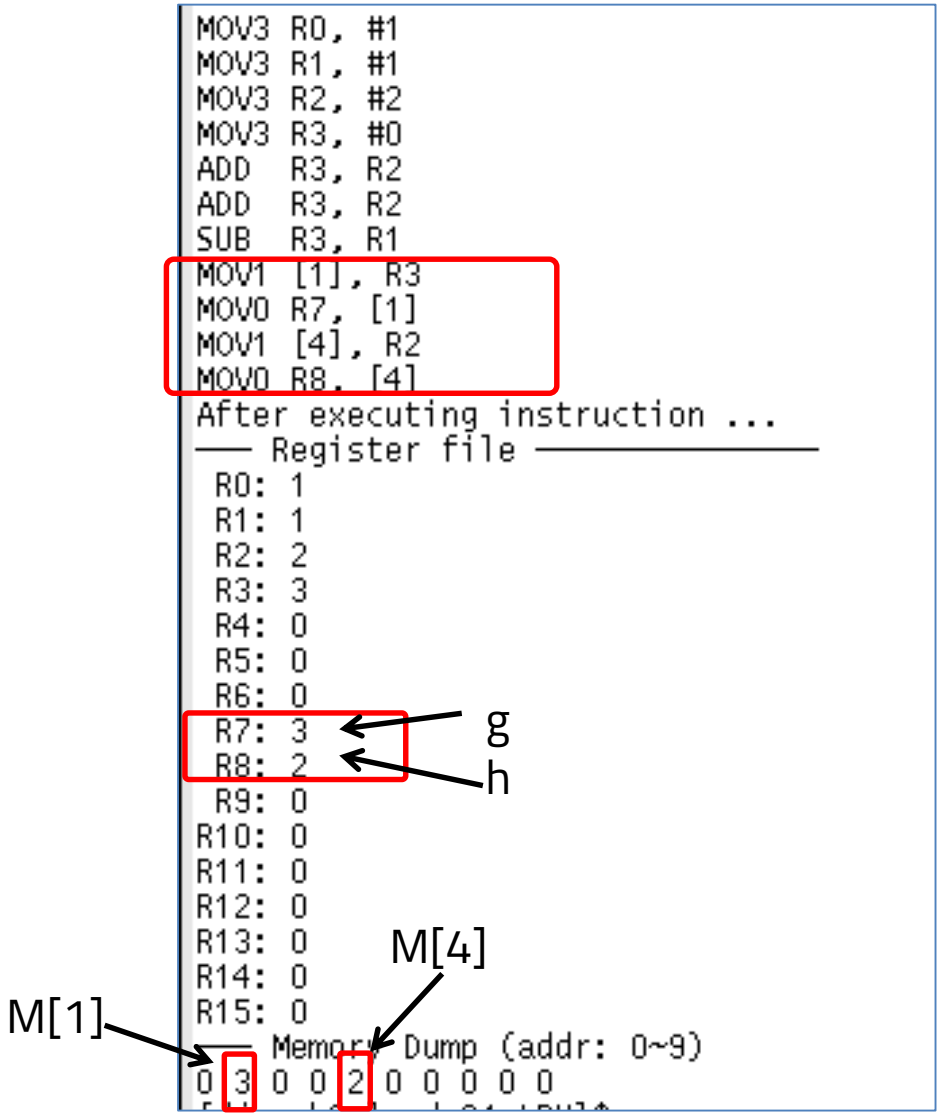
코드실행 라인수



file.bin

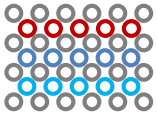
```
0011000000000001
0011000100000001
0011001000000010
0011001100000000
0100001100100000
0100001100100000
0101001100010000
0001001100000001
0000011100000001
0001001000000100
0000100000000100
```

TPU 프로세서 코드 실행 결과



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)