uprocessor App.

Introduction to uProcessor & ARM Architecture

목차

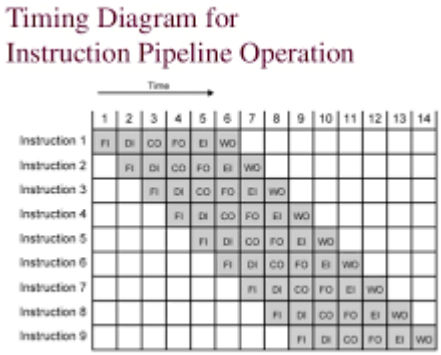
1. Instruction Set Architecture

```
Amax = A[0];
for(i=1; i!=4; i++){
    if(Amax<A[i])
        Amax=A[i];
}
```

```
                                start   end
        lw      $t0, 0($s1)         $t0[A]->[A]
        addi    $t1, $zero, 0       $t1[0]->[loop time]
loop :  add     $t1, $t1, 1         $t2[x]->[A+4*loop time]
        beq     $t1, $s2, done      $t3[x]->[A[loop time]]
        add     $t2, $t1, $t1       $t4[x]->[0 or 1]
        add     $t2, $t2, $t2       $s1[A]->[A]
        add     $t2, $t2, $s1       $s2[A_end]->[A_end]
        lw      $t3, 0($t2)
        slt     $t4, $t0, $t3       $t2->address
        beq     $t4, $zero, loop    $t3->value
        addi    $t0, $t3, 0
        j       loop
done :  ...
```

2. Microarchitecture

- A set of steps that a processor takes to execute a particular ISA
- Microarchitecture features
    - Cache memory : 메인 메모리와 통신 횟수 감소. 속도 증가
    - Pipelineing : 같은 명령어를 더 빠른 속도로 실행 가능



        - w/ pinelining
    - Out-of-Order execution
        - 임의로 순서를 바꿔서 계산할 수 있도록 해주는 프로그램
    - Superscalar Issue
        - 한 cycle에 여러 instruction을 수행하도록 우겨넣는 프로그램

1. fetch : instruction 가져옴
2. decode : 해석
3. execute : 실행
4. write : register에 저장

3. Arm Architecture

- User's View(ISA) : 사용자에게 할당
    - Instruction set
    - Memory management table structures
    - Exception handling model etc.
- Organization(Micro Architecture) : 보드 자체의 성능
    - Pipelin structure
    - Cache
    - Table-walking hardware
    - Translation look-aside buffer etc.

4. Cortex-A9 Processor

- Thumb(16bit) & Arm(32bit) instruction set
    - 각자 다른 회로로 해석한다고 추정
- Advanced SIMD architecture,***NEON***, for multimedia application.
    - 이후 NOEN을 직접 설계할 수도 있음.
    - in normal method [a1] + [b1] = [c1] -> add 1 [a2] + [b2] = [c2] -> add 2 [a3] + [b3] = [c3] -> add 3
        - 3 add instruction to calculate
    - in NEON [a1] + [b1] = [c1] [a2] + [b2] = [c2] -> vadd1 [a3] + [b3] = [c3]
        - 1 vadd instruction to calculate

ps. SIMD & Superscalar

- SIMD : instruction set 차이 有, 즉 특별한 명령어를 사용
- Superscalar : instruction set 차이 無, 여러 instruction을 한 사이클에 우겨넣는다 이해하면 됨.

5. Data Types & Arithmetic

- Data types in memory
  - Byte(8 bits)
  - Halfword(16 bits)
  - Word(32 bits)
  - Doubleword(64 bits)

```
S1[0x0]         ->byte
S2[0x00]        ->half word
S3[0x0000]      ->word
S4[0x0000 0000] ->double word
```

- Data types in Register(32bit)
  - One 32-bit pointer
  - One unsigned/signed 32-bit integer
  - One unsigned 16-bit/8-bit integer(zero-ext.)
    - 32bit 기준인데 어째서 하나?
  - One signed 16-bit/8-bit integer(sign-ext.)
    - 32bit 기준인데 어째서 하나?
  - Two 16-bit integers(short등)
  - Four 8-bit integers
  - Half of 64-bit unsigned/signed integer

```
1 [0x1234] -> pointer
2 [0x5678] -> 32-bit ingeter
3 [0x12][0x34]
4
5
6 [0xFF][0xFF] -> two 16-bit integers
7 [0xF][0xA][0x1][0xB] -> four 8-bit integers
8 [0x1234]
9 [0x5678]  ->half 64 bit integer
```