uprocessor App.

ARM Architecture

목차

1. ARM Processor Mode

- mode
    - User : 일반적인 모드
    - FIQ : Fast Interrupt
    - IRQ : Interrupt
    - etc..
    - difference between polling, interrupt and exception?
        - polling : 프로그램이 상시로 값들을 참조하여 사건이 발생했는지 확인. 프로그램이 중간에 확인하는 시간이 낭비된다는 단점이 있음.
        - interrupt : 하드웨어 상에서 사건이 발생하면 사건이 발생했다는 신호를 줌.
        - exception : 소프트웨어 상에서 사건이 발생하면 사건이 발생했다는 신호를 줌

2. ARM Core Register

- 같은 이름의 register이라도 mode에 따라서 물리적인 위치가 다르다.
    - why?
    - SP : Stack Pointer
    - LR : Link Register
    - PC : Program Counter
    - APSR : Program Status Register
        - CPSR : Current Program Status Register
        - SPSR : Saved Program Status Register
        - mode가 바뀔때 CPSR을 SPSR에 저장.
- PSR(Program Status Register)
    - Condition flags
        - N = Negative result from ALU(set if negative)
        - Z = Zero result from ALU(set if zero)
        - C = ALU operation carried out(set if carried out)
        - V = ALU operation overflowed(set if over/underflow)

- Mode field : Current mode of processor

3. Exceptions

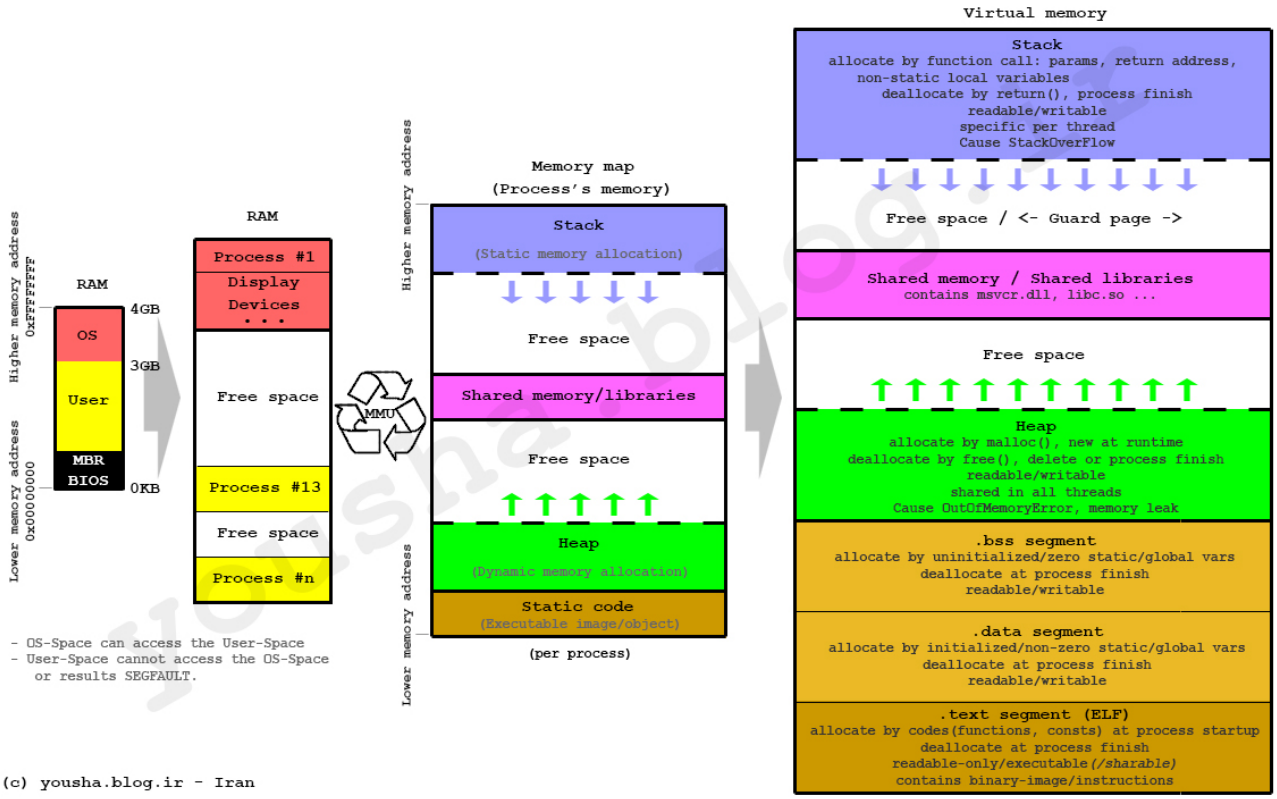
- definition : **conditions or system events** that requires **remedial action** or an update of system status by **privileged software**
- when execution occurs, the system halts normal execution and instead executes dedicated software routine(like interrupt service routine in interrupt)
- interrupt : hardware detects unexpected condition
- exception : processor detects unexpected condition
- Exception entry
    1. Copies CPSR into SPSR
    2. Set appropriate CPSR bits
        - state, endianness, mode, interrupt etc.
    3. Stores return address in LR
    4. Sets PC to vector address
    5. Retrieve Vecter Table and go to software routine
    6. Handle the exception
    7. Restore data
- Exception Types
    - Reset
    - Undefined
    - Prefetch/data abort
    - Interrupt(IRQ,FIQ)
- Vector Table
    - One entry per exception type
    - One intruction at each entry
    - Branch instruction except for FIQ
    - FIQ placed the handler itself directly, so it react faster

4. Memory Model

```
0x0000 0000  [       ]
             [       ]  ->Uncached/Peripherals
    .
    .
    .
             [       ]
             [       ]  ->Privileged Access/OS
             [       ]  ->Application Model/User Access
             [       ]
    .
```

```
      .
      .
              [       ]
0xFFFF FFFF   [       ]  ->Cached/Read-only/Vectors
```

- Memory Types
  - Controls : 1. Memoryaccess ordering rules 2. Caching/buffering behavior
  - Mutually exclusive memory types supported :
    1. Normal : data, instruction
    2. Device : device / peripherals, data
    3. Strongly-ordered : device / peripherals, data
- Memory Hierarchy



  - vitural memory?
  - storage vs memory?

5. Cache

- L2 Cache : External & von-Neumann model
  - von-Neumann model : instrument memory = data memory
  - 0KB to 1MB
  - 8 words per line
- L1 cache L Integrated & Harvard
  - Harvard mode : instrument memory와 data memory가 구분
  - 16KB/32KB/64KB
  - 4-way set-associative
  - 8 words per line

6. Set Associative Cache

```
    Main Memory             Cache way 0
                 call                           |1|2|3|4|
0x0000.0000[    ] 1    0[   ][   ][   ][   ]    |1|1|2|3|
0x0000.0010[    ] 4    1[   ][   ][   ][   ]    | | | |4|
0x0000.0020[    ]      2[   ][   ][   ][   ]    | | | | |
0x0000.0030[    ]      3[   ][   ][   ][   ]    | | | | |
0x0000.0040[    ] 2
0x0000.0050[    ]           Cache way 1
0x0000.0060[    ]      0[   ][   ][   ][   ]    | |2|2|2|
0x0000.0070[    ]      1[   ][   ][   ][   ]    | | | | |
0x0000.0080[    ] 3    2[   ][   ][   ][   ]    | | | | |
0x0000.0090[    ]      3[   ][   ][   ][   ]    | | | | |
```

- Cache의 way에 따라 규칙이 변경.

```
   Address
[                  ] [         ] [   ] [  ]
        tag           set(=Index) word byte
       19bit              8bit    3bit 2bit

만약 int *p의 데이터를 가져온다 가정해보자.
integer -> 4byte = 32bit
Q1. 주변 데이터는 같은 라인에 저장? 혹은 다른 라인에 저장?

Q1-1. 만약 같은 라인에 저장된다고 한다면, p[1]의 메모리 주소는 p+4가
된 다. 이는 cache 상에서 그 아래 line에 해당되는것이 아닌가?

Q2. 만약 cache memory의 개수가 늘어나면 byte에서 비트를 끌어다
쓰는가?
```

- tag : 메인 메모리 상의 주소
- set(index) : 하나의 Cache Memory의 라인
- word : 몇번째 cache memory인지 나타내주는 곳
- byte : 남는 잉여 데이터