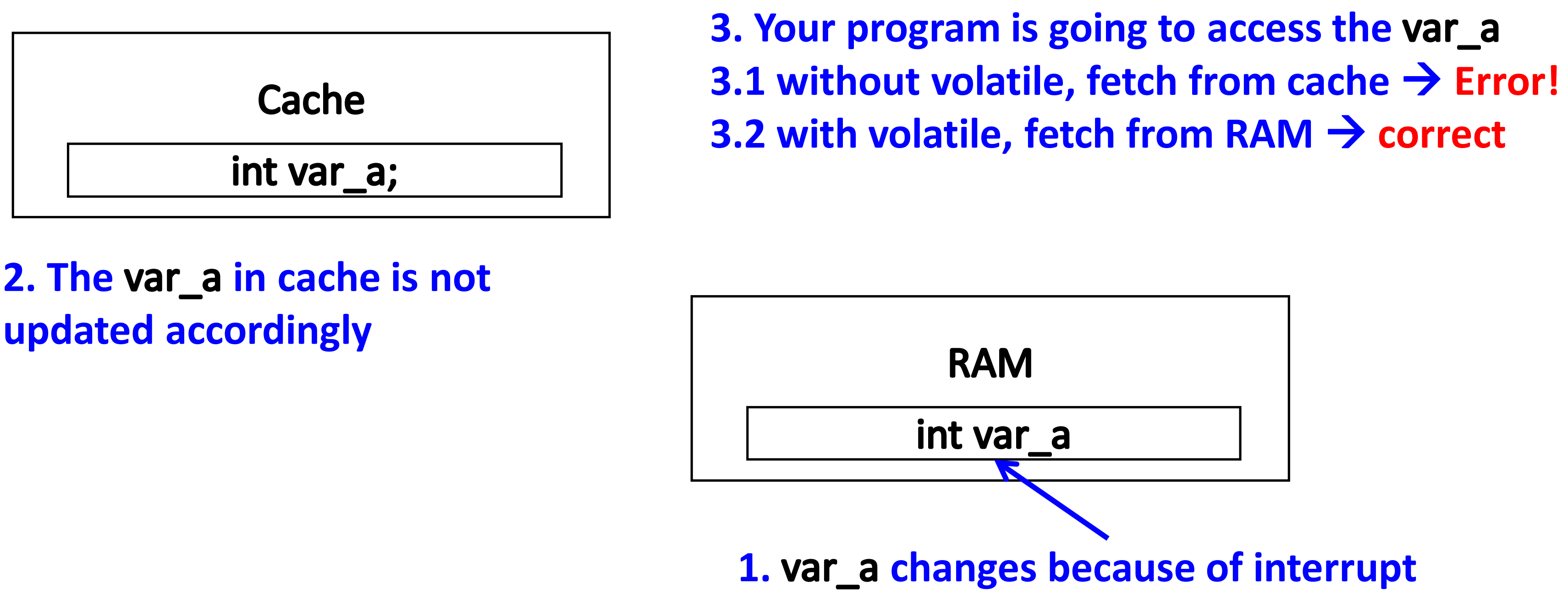
volatile

- Usage → volatile int var_a;
- Read data/variable from physical memory space instead of cache
- Tell the compiler do not do optimization on this variable



3. Your program is going to access the var_a
3.1 without volatile, fetch from cache → Error!
3.2 with volatile, fetch from RAM → correct

2. The var_a in cache is not updated accordingly

1. var_a changes because of interrupt

const

- Take a variable as a constant (cannot change its value)
- **const** int a = 30;
- **const** int a[5] = {1, 2, 3, 4, 5};
- **const** int *p = a   (use as function argument)
  - p is a pointer that direct to a constant (you cannot change the content of a)
  - *p = 3 → Error !
- int * **const** p = a   (use as function argument)
  - The location that p points to is not changeable
  - p = b → Error, p++ → Error
- **const** int * **const** p = a   (use as function argument)
  - Have the characteristics of the above two

static/extern

- static
  - Static function → can not call by the procedure located in other file
    - static int swap (int *a, int *b);
  - Static variable
    - Declare inside a function → always exists
    - Declare outside a function → A global variable but cannot change by the procedure in other file
- extern
  - in 1.c → int var_a;
  - in 2.c → extern int var_ a;
  - You can use the same var_a variable in 2.c (1.c and 2.c share the common var_a)

extern

```
#include <stdio.h>

extern int b;

int main (void)
{
   int a = 10;
   int *p;

   p = &a;
   printf ("*p= %d \n", *p);

   changeP(&p);
   printf ("*p= %d \n", *p);

   p = &b;
   b = 2000;
   printf ("*p= %d, p: %x \n", *p, p);
}
```

```
in change.c

#include <stdio.h>

int b = 100;

void changeP (int **pp)
{
   *pp = &b;
   **pp = 1000;
   printf ("changeP: %x\n", &b);
}
```

```
*p= 10
changeP: 80495e4
*p= 1000
*p= 2000, p: 80495e4
```

union

```
#include <stdio.h>

union StateMachine {
    char character;
    int number;
    char *str;
};
```

```
int main(void) {

    union StateMachine machine;

    machine.number = 1;
    printf("sizeof: %d\n", sizeof(machine));
    printf("number: %d\n", machine.number);

    return 0;
}
```

http://caterpillar.onlyfun.net/Gossip/CGossip/union.html

union

```
#include <stdio.h>

#defined NOT_SEL 0xFF
typedef unsigned char bool;

typedef struct stu {
    int ID;
    int mathScore;
    union
    {
        bool selected;
        int hisScore;
    } his;
} tStu;
```

```
int main(void) {
    tStu stu;

    stu.ID = 1;
    stu.mathScore = 90;

    stu.his.hisScore = 20;
 // stu.his.selected = NOT_SEL;

    printf("sizeof: %d\n", sizeof(stu.his));
    printf(“selected: %d\n",
                  stu.his.selected); //the result?

    return 0;
}
```

- Union is commonly integrated in a structure
- The content of the union may be different types of internal signals in an OS

enum

http://caterpillar.onlyfun.net/Gossip/CGossip/enum.html

- Declaration
  - **enum Action {stop, sit, stand, walk, run};**
  - **enum Action {stop = 1, sit, stand, walk, run};**

  - **enum Action {stop = 1, sit, stand=2, walk, run};**
    - **sit and stand will be both 2**

- Usage
  - **enum Action action = stop;**

```c
#include <stdio.h>

typedef enum test
{
   #include "enumm.h"
   NUMBER
} testEnum;

int main (void)
{
    int a = 10;
    int *p;
    testEnum x = 5;

    if (x >= NUMBER)
        printf (“Larger than %d\n“, NUMBER);
    else
        printf (“OK! NUMBER: %d \n", NUMBER);
    return 0;
}
```

enumm.h

```c
stop = 0,
sit,
walk,
run,
stand,
```

W17-assignment

- Write a program to allow user to enter “Name” and “Phone number”
    - Store the name by char name[10]
    - Phone number may have two types: “home” or “cellular” (integrated by union)
        - For the home number, you should record the area code and number
        - For the cellular phone number, you should record operator’s name by enum (CHT, FET, TWN) and number
- The inputted information should be maintained by a linked list
- After finishing enter a new user, you should print all information