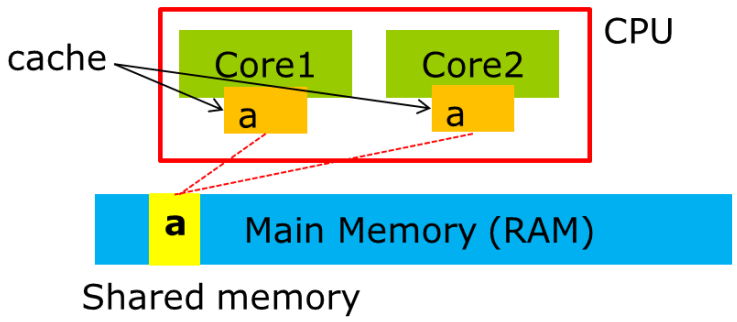
Debates/Discussions – Week 4

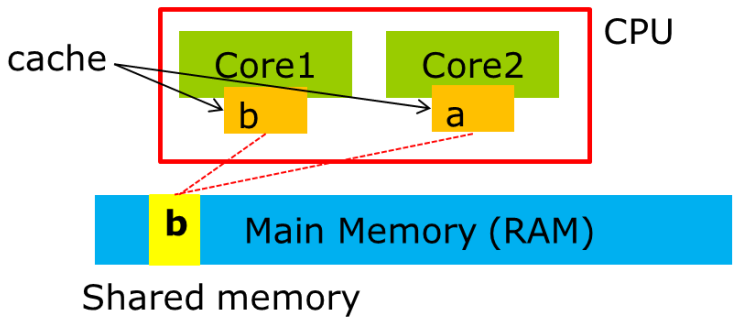
(1) When does a process exit from the running state? Give three examples.

(2) What is a process context? Describe the actions taken by kernel to context-switch between processes.

(3) Solve question 2 and question 3 in the next slides. ***getpid()*** system call returns the *process id* (pid) of caller process.

(4) Explain why the **cache coherence problem** happens when using shared memory for multicore CPUs





Question 2

Using the program in Figure, identify the values of **`pid`** at lines A, B, C, and D. (Assume that the actual **`pids`** of the parent and child are **`2600`** and **`2603`**, respectively.)

recall

`fork()` return value:

- Child process: 0
- Parent process: **`pid`** of child process (>0)

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
pid_t pid, pid1;

    /* fork a child process */
    pid = fork();

    if (pid < 0) { /* error occurred */
       fprintf(stderr, "Fork Failed");
       return 1;
    }
    else if (pid == 0) { /* child process */
       pid1 = getpid();
       printf("child: pid = %d",pid); /* A */
       printf("child: pid1 = %d",pid1); /* B */
    }
    else { /* parent process */
       pid1 = getpid();
       printf("parent: pid = %d",pid); /* C */
       printf("parent: pid1 = %d",pid1); /* D */
       wait(NULL);
    }

    return 0;
}
```

Figure 3.29 What are the pid values?

Question 3

What is the output of line A?

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int value = 5;

int main()
{
pid_t pid;

   pid = fork();

   if (pid == 0) { /* child process */
      value += 15;
      return 0;
   }
   else if (pid > 0) { /* parent process */
      wait(NULL);
      printf("PARENT: value = %d",value); /* LINE A */
      return 0;
   }
}
```

Figure 3.30 What output will be at Line A?

`fork()` return value:

- Child process: 0
- Parent process: **`pid`** of child process (>0)