April 9 ~ April 13 / 2018
HITBSecConf
Keeping Knowledge Free for Over a Decade

Hiding Tasks via Hardware Task Switching

[Kyeong Joo Jung – Ajae.dll]

This research was funded by the MSIT, Korea, under the "ICT Consilience Creative Program" (IITP-2017-R0346-16-1007) supervised by the IITP, by the KEIT, Korea, under the "Global Advanced Technology Center" (10053204), and by the NRF, Korea, under the "Basic Science Research Program" (NRF-2015R1C1A1A01053788).

Ajae.dll
Stony Brook University
SUNY Korea

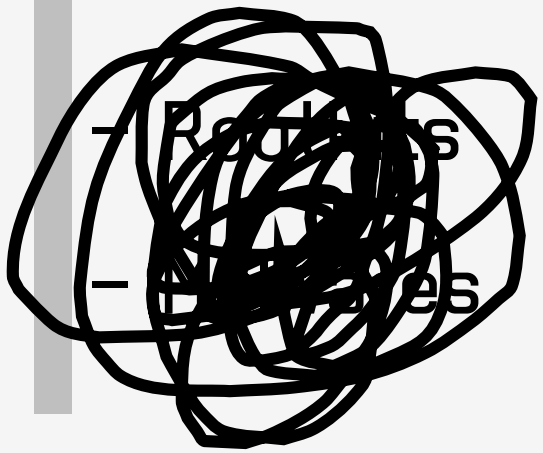
Ajae.dll

차세대 보안리더 양성 프로그램
BEST OF THE BEST

kyeongjoo.jung@stonybrook.edu

CONTENTS

Hiding Tasks?



- Task Switching Method

H/W
Task Switching

- Used until Windows 3.1 (~1993)
- Uses structures defined by CPU Manufacturer
- Access to CPU directly

S/W
Task Switching

- Used from Windows NT (1994~)
- Uses structures defined by OS
- Access CPU through OS Scheduler

H/W
Task Switching

- Used until Windows 3.1 (~1993)
- Uses structures defined by CPU Manufacturer
- Access to CPU directly

S/W
Task Switching

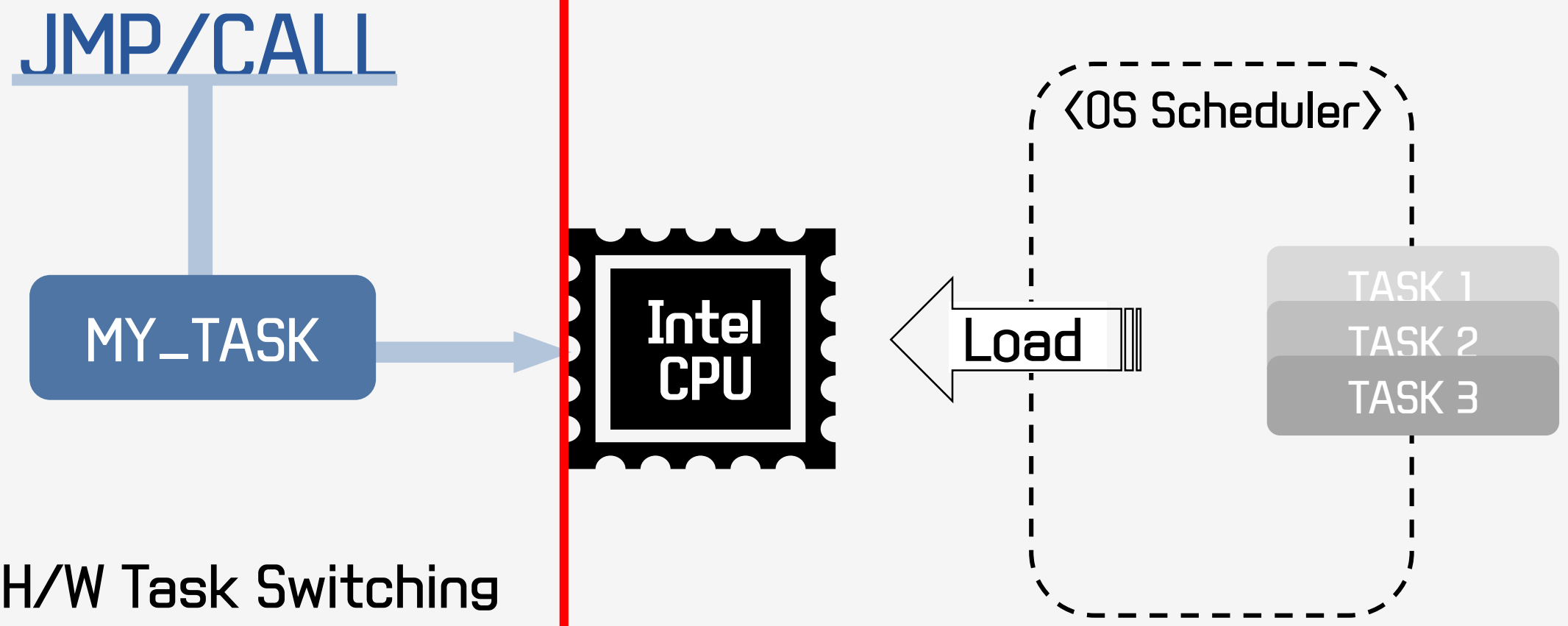
- Used from Windows NT (1994~)
- Uses structures defined by OS
- Access CPU through OS Scheduler

> S/W
> Task Switching

- Used from Windows NT (1994~)
- Uses structures defined by OS
- Access CPU through OS Scheduler

> H/W
> Task Switching

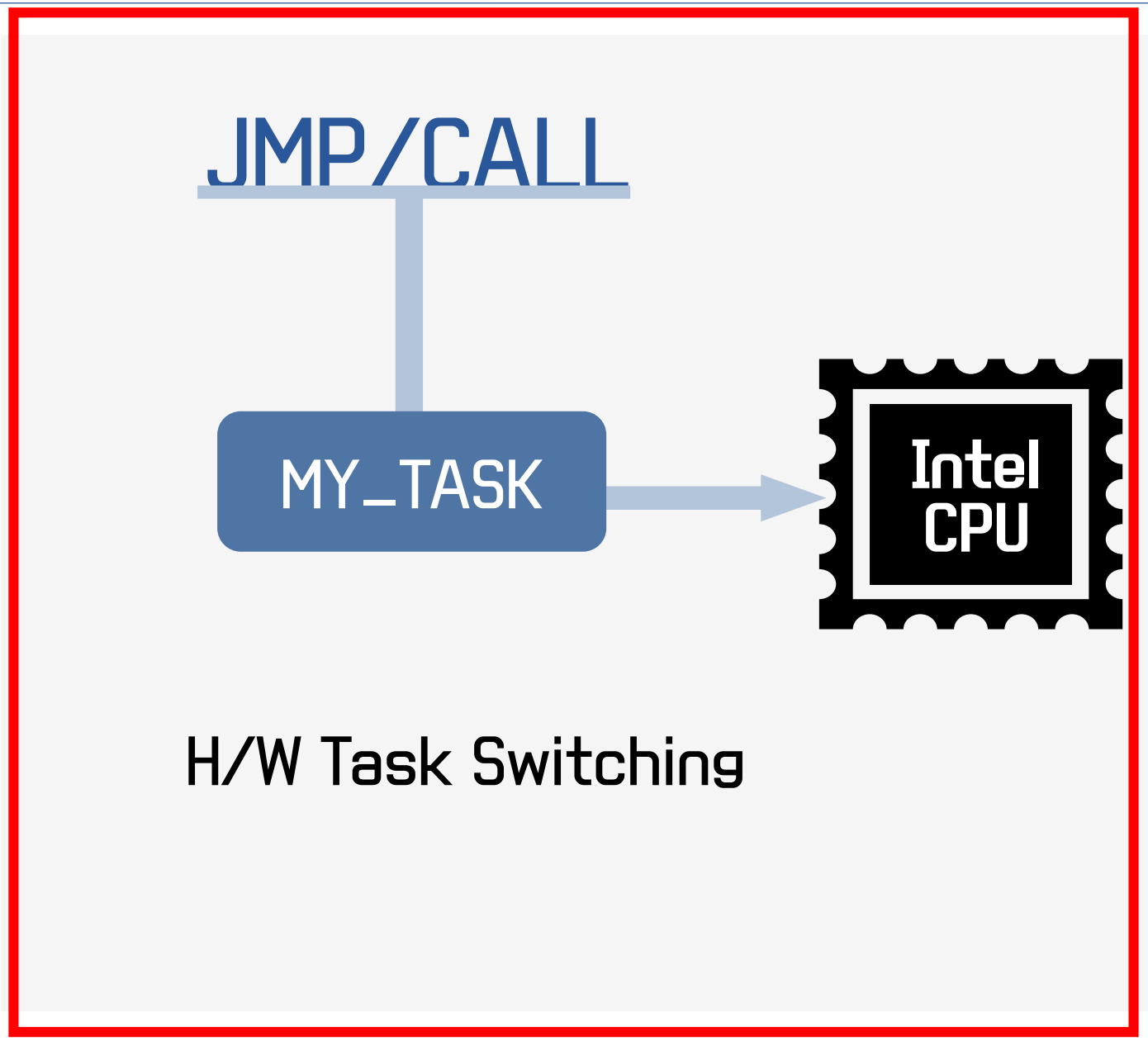
1. Intro



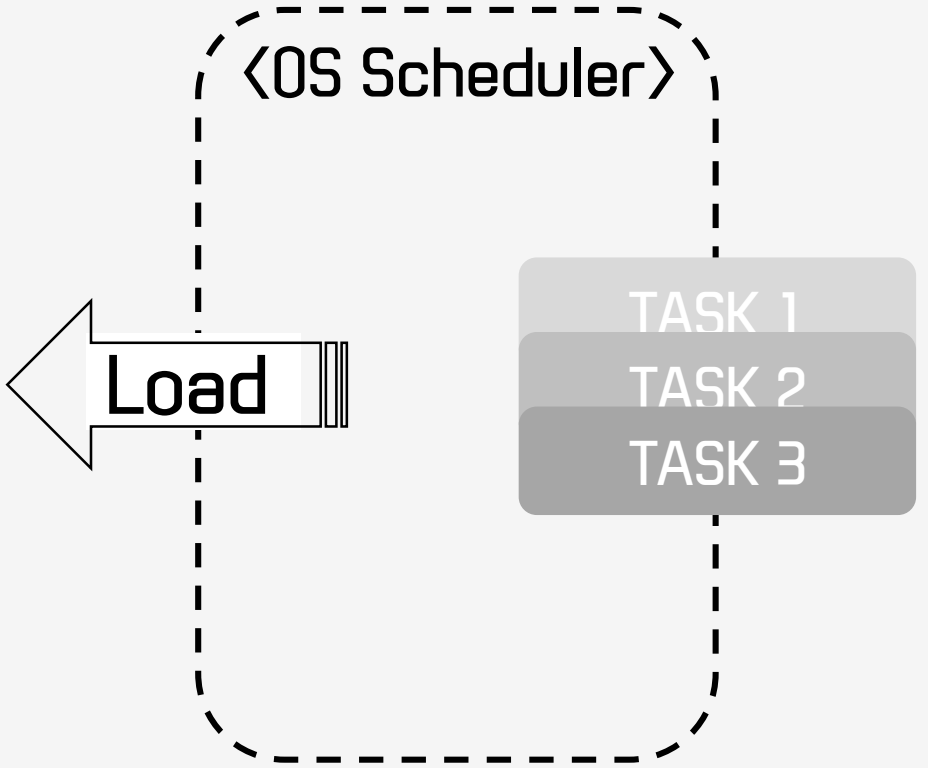
H/W Task Switching

S/W Task Switching

1. Intro



H/W Task Switching



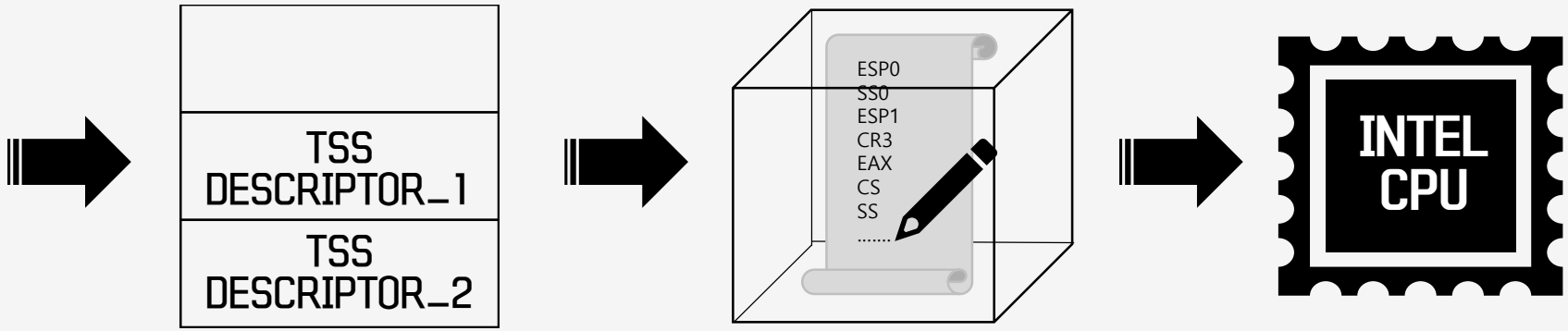
S/W Task Switching

1. Intro

2. Explanation

H/W Task Switching?

JMP / CALL

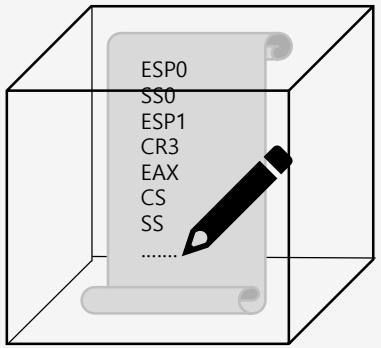


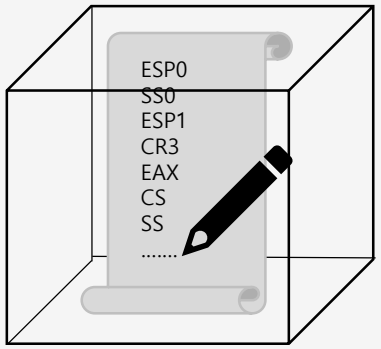
〈Trigger〉 〈Global Descriptor Table〉 〈Call TSS〉 〈Operate CPU〉

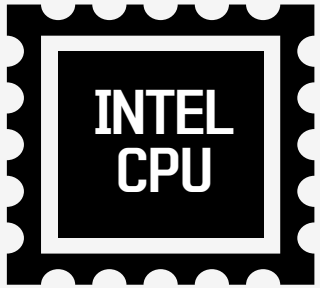
- Task State Segment
- Defined by CPU manufacturer
- Saves states of Task

TSS

JMP / CALL







〈Trigger〉

〈Global Descriptor Table〉

〈Call TSS〉

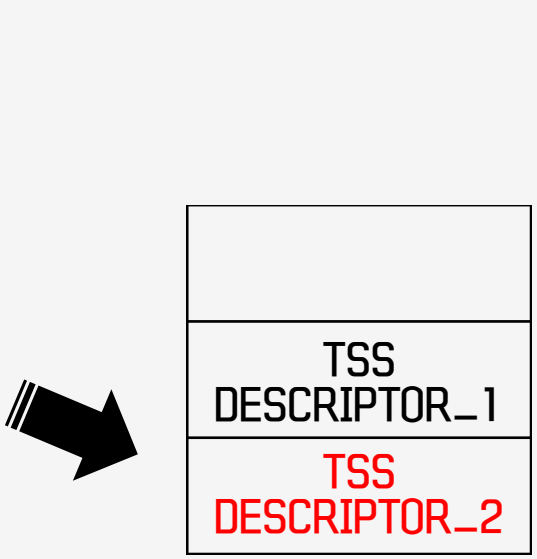
〈MY_TSS〉

〈Operate CPU〉

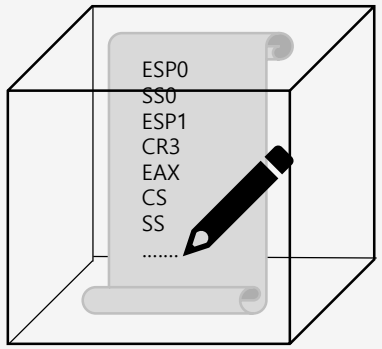
H/W Task Switching

JMP / CALL

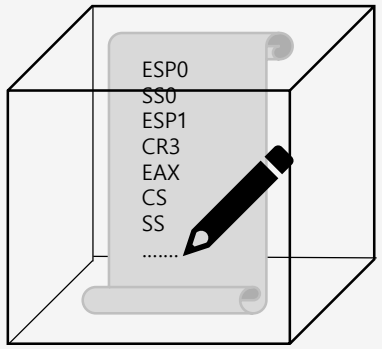
<Trigger>



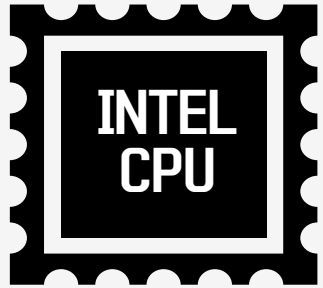
<Global Descriptor Table>



<Call TSS>



<MY_TSS>



<Operate CPU>

TSS Descriptor

GDT INDEX	Descriptor Name & State
0x28	**32-bit TSS (Busy)**
0x50	32-bit TSS (Available)
0x58	32-bit TSS (Available)
0xA0	32-bit TSS (Available)

GDT INDEX	Descriptor Name & State
0x16	**32-bit TSS (Busy)**
0x31	32-bit Double Fault TSS (Available)

Modifying TSS Descriptor

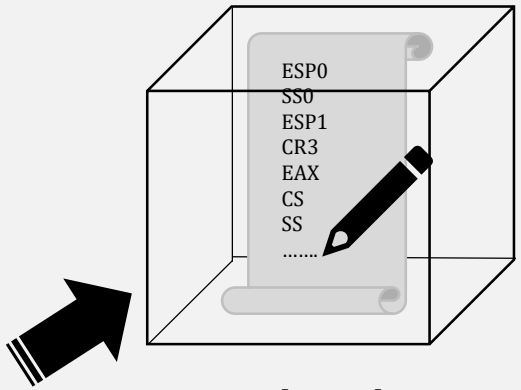
GDT INDEX	Descriptor Name & State
0x28	32-bit TSS (Available)
0x50	**32-bit TSS (Busy)**
0x58	32-bit TSS (Available)
0xA0	32-bit TSS (Available)

GDT INDEX	Descriptor Name & State
0x16	32-bit TSS (Available)
0x31	**32-bit Double Fault TSS (Busy)**

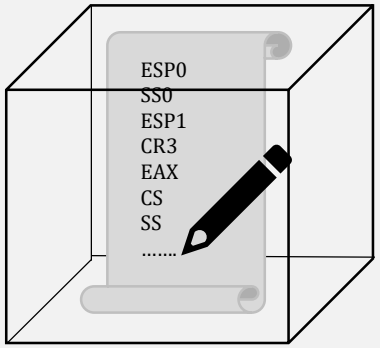
3. Result

Result

INDEX	DESCRIPTOR
0x28	TSS (Busy)
0x50	MY_TSS (Available)
0x58	TSS (Available)
0xA0	TSS (Available)



〈TSS〉



〈MY_TSS〉

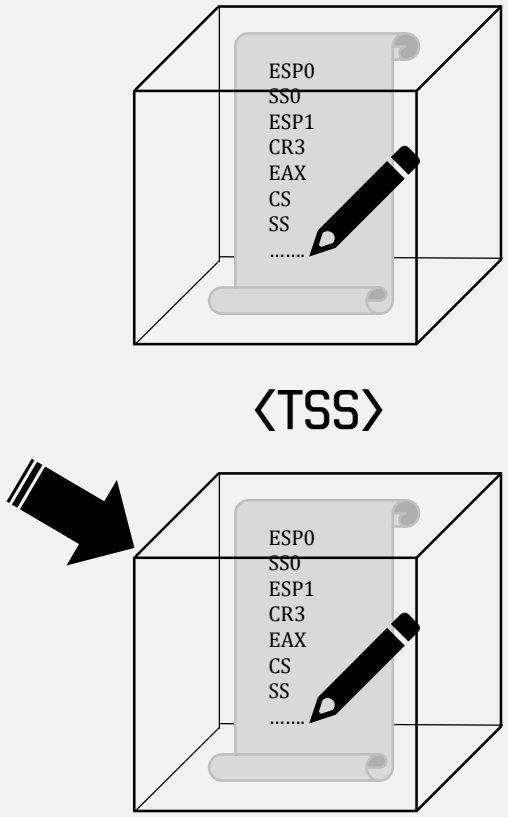

```
[Before Task Switching]
---------[After Task]
---------_SHOW_ME_THE_AJAEDLL
---------[After][Success calling for '2' Times]
[Before][Success calling for '2' Times]

[Before Task Switching]
---------[After Task]
---------_SHOW_ME_THE_AJAEDLL
---------[After][Success calling for '3' Times]
[Before][Success calling for '3' Times]
```

3. Result

Printing

INDEX	DESCRIPTOR
0x28	TSS (Available)
0x50	MY_TSS (Busy)
0x58	TSS (Available)
0xA0	TSS (Available)



〈TSS〉

〈MY_TSS〉

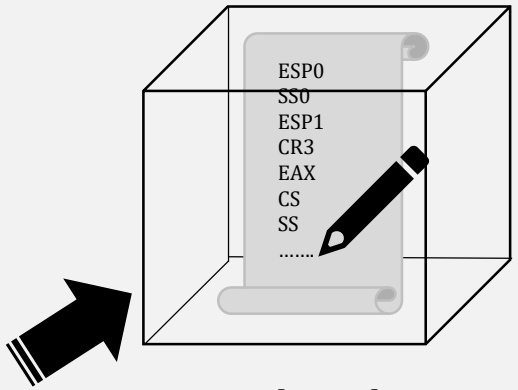

```
[Before Task Switching]
---------[After Task]
---------_SHOW_ME_THE_AJAEDLL
---------[After][Success calling for '2' Times]
[Before][Success calling for '2' Times]

[Before Task Switching]
---------[After Task]
---------_SHOW_ME_THE_AJAEDLL
---------[After][Success calling for '3' Times]
[Before][Success calling for '3' Times]
```

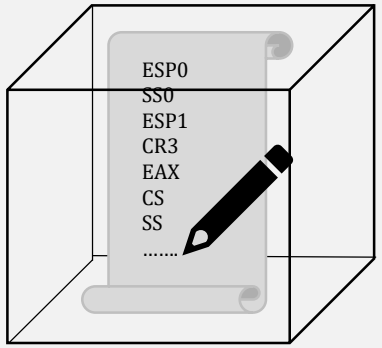
3. Result

Printing

INDEX	DESCRIPTOR
0x28	TSS (Busy)
0x50	MY_TSS (Available)
0x58	TSS (Available)
0xA0	TSS (Available)



〈TSS〉



〈MY_TSS〉


```
[Before Task Switching]
---------[After Task]
---------_SHOW_ME_THE_AJAEDLL
---------[After][Success calling for '2' Times]
[Before][Success calling for '2' Times]

[Before Task Switching]
---------[After Task]
---------_SHOW_ME_THE_AJAEDLL
---------[After][Success calling for '3' Times]
[Before][Success calling for '3' Times]
```

3. Result

Printing Results


```
[Before Task Switching]
---------[After Task]
---------_SHOW_ME_THE_AJAEDLL
---------[After][Success calling for '2' Times]
[Before][Success calling for '2' Times]

[Before Task Switching]
---------[After Task]
---------_SHOW_ME_THE_AJAEDLL
---------[After][Success calling for '3' Times]
[Before][Success calling for '3' Times]
```


```
[Before Task switching] --------------
---------------[After Task switching]
---------------_SHOW_ME_THE_AJAEDLL_
---------------[After][Success calling for '1' Times]
[Before][Success calling for '1' Times]
[Before Task switching] --------------
---------------[After Task switching]
---------------_SHOW_ME_THE_AJAEDLL_
---------------[After][Success calling for '2' Times]
[Before][Success calling for '2' Times]
[Before Task switching] --------------
---------------[After Task switching]
```

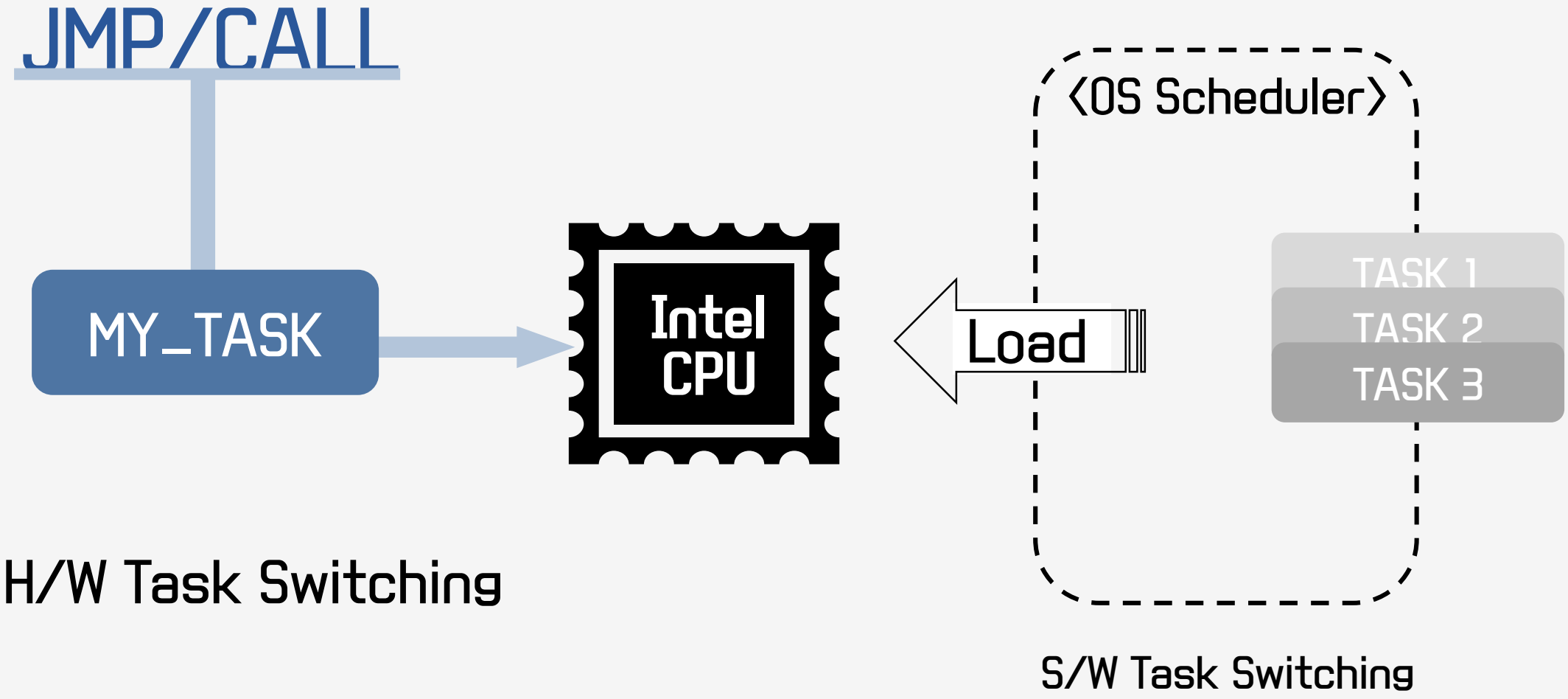
CPU Usage Video

[redacted]

Detection method	Tool	Detection
Signature based detection	Icesword	X
	Malwarebytes anti-rootkit	X
	Sophos Vrius Removal Tool	X
	TDSSKiller	X
Behavior based detection	Alyac (Korean program)	X
Integrity based detection	Afick	X
Hooking detection	Gmer	X
	Radix	X

[redacted]

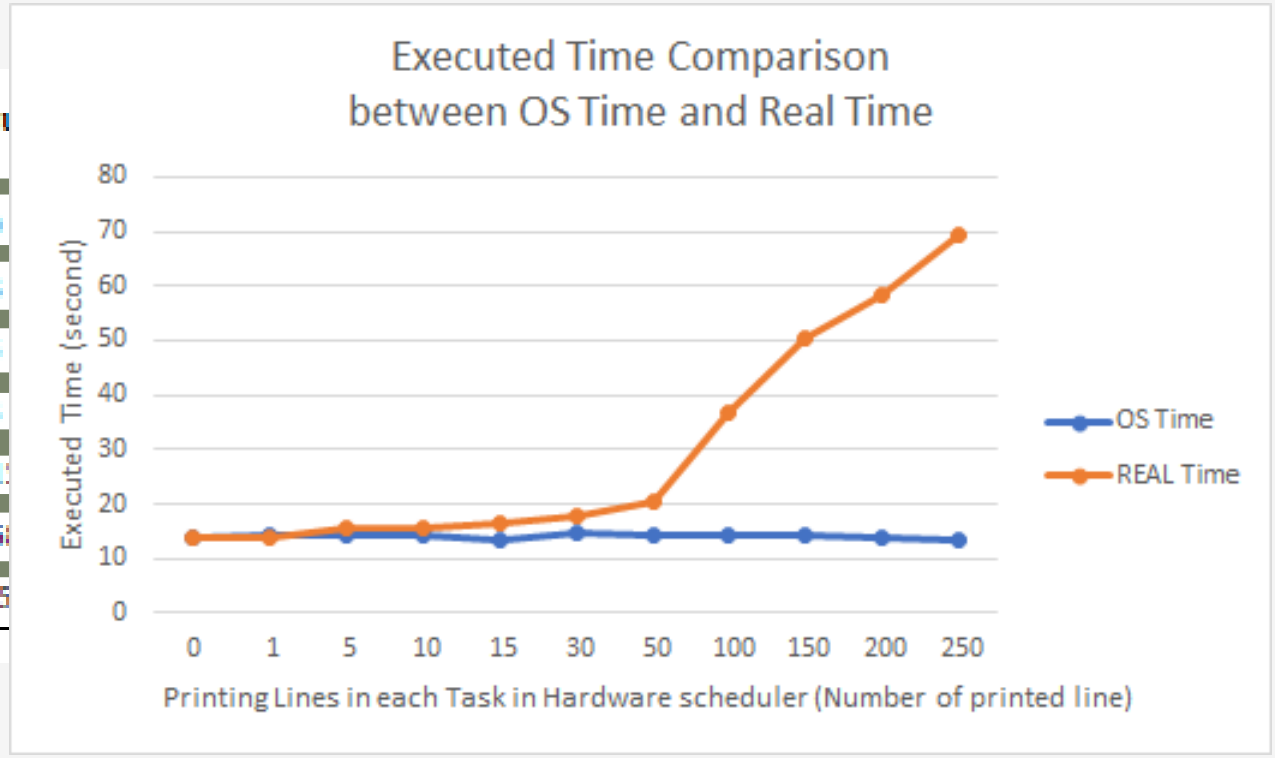
3. Result


JMP/CALL
MY_TASK
Intel CPU
H/W Task Switching
〈OS Scheduler〉
Load
TASK 1
TASK 2
TASK 3
S/W Task Switching

- 10 cases : increasing CPU utilization
- User will be looking at 55% only
- Sum of H/W and OS = actual required CPU usage
- User waits -> suspicious -> check

Table 1: Result of task execu

Number of printed lines in task in hardware	0	1	5	10
CPU utilization of task in hardware (%)	0	9	14	18
CPU utilization of task in OS (%)	55	55	55	55
Required CPU utilization of both tasks (%)	55	64	69	73
Execution time of task using OS time (sec)	13.99	14.03	14.44	14.1
Execution time of task using online (sec)	13.96	13.99	15.56	15.5
Time difference of both time (sec)	-0.04	-0.04	1.12	1.45



4. Conclusion

Problem

1. Gives chance to attackers
- Servers, zombie PCs
- Mining

2. Belief in OS
- OS event log, time
- Investigation

-> Power of ring 0

Limitation

1. 32 bit only yet
- Still many out there
- Working on 64 bit!!

2. Memory scan
- Normal users won't scan it
- Don't make it suspicious

4. Conclusion

Contribution

1. New approach of hiding tasks
2. Potential Danger
   - Accessing files, network, CPU..
3. Ring 0 powerful than we think

Thank you!

Q&A

- AJAE.DLL -

kyeongjoo.jung@stonybrook.edu