


Administrative matter


- Final “class project” Lab
 - It is a Lab, not homework
 - Outside lab work not expected, nor sought for
 - Partial credit will be given
 - Depend on how much you get to work
 - It is not expected that everyone will get everything to work
- Extra lab hour
 - Today, 9:00PM-10:00PM
 - You MUST turn in your prototype by 10PM



1

Administrative matter

- Homework #6 Due now
- Final examination
 - Tuesday 12/10, 8-11AM
 - In classroom
 - Cover ESD 8,10,11
 - Cover RTSPL 5, 13.1-13.11
 - Cover all labs



2

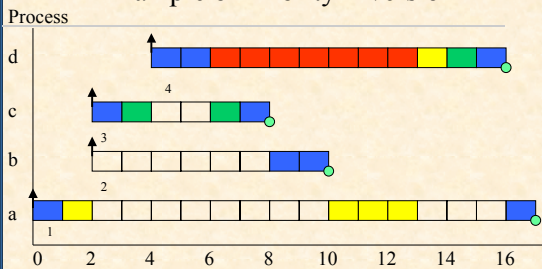
Priority Inversion

- Consider the executions of four periodic processes
 - a, b, c and d; and two resources: Q and V

Process	Priority	Execution Sequence	Release Time
a	1	EQQQVE	0
b	2	EE	2
c	3	EVVE	2
d	4	EEQVE	4

3

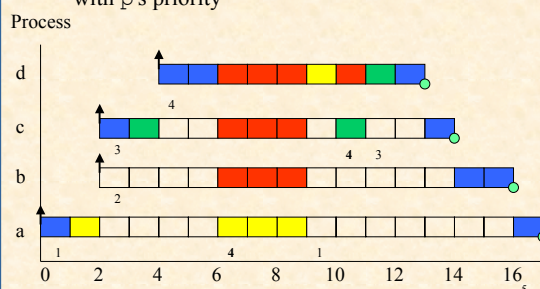
Example of Priority Inversion



4

Priority Inheritance

- If process p is blocking process q, then q runs with p's priority



5

Calculating Blocking

- If a process has k critical sections
 - then the maximum number of times it can be blocked is k
- If B is the maximum blocking time and K is the number of critical sections,
 - the process i has an upper bound on its blocking given by:

$$B_i = \sum_{k=1}^K usage(k, i)C(k)$$

- Under the priority inversion scheme

6

Blocking

- Rule to calculate “usage”
 - “1 if resource k is used by at least one process with a priority less than i, and at least one process with a priority greater or equal to i, 0 otherwise”
 - Or,
 - Resources which are accessed by processes equal or higher than i,
 - and, are accessed by processes lower than i

$$B_i = \sum_{k=1}^K usage(k, i)C(k)$$



Calculate Blocking

Q used by a, d
V used by c, d

Ba=0 ; Bb=4 ; Bc=4 ; Bd=4+2=6

Process	Priority	Execution Sequence	Release Time
a	1	EQQQQE	0
b	2	EE	2
c	3	EVVE	2
d	4	EEQVE	4



Response Time and Blocking

$$R_i = C_i + B_i + I_i$$

$$R_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

$$w_i^{n+1} = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{T_j} \right\rceil C_j$$



Priority Ceiling Protocols

Two forms

- Original ceiling priority protocol
- Immediate ceiling priority protocol



On a Single Processor

- CPPs guarantee
 - A high-priority process can be blocked at most once during its execution by lower-priority processes
 - Deadlocks are prevented
 - Transitive blocking is prevented
 - Mutual exclusive access to resources is ensured
 - by the protocol itself

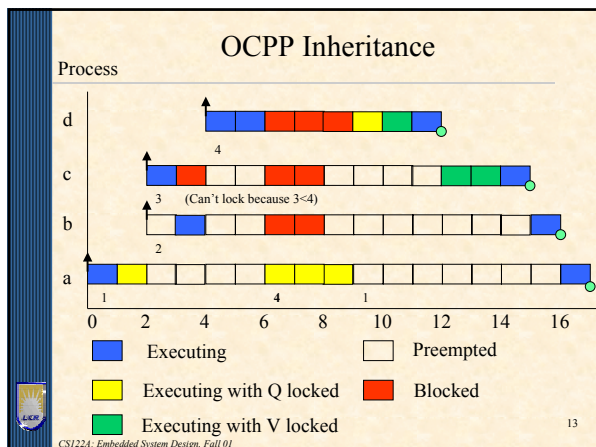


OCPP

- Each process has a static default priority assigned
 - perhaps by the deadline monotonic scheme
- Each resource has a static ceiling value defined
 - this is the maximum priority of the processes that use it
- A process has a dynamic priority that is
 - the maximum of its own static priority and
 - any it inherits due to it blocking higher-priority processes.
- A process can only lock a resource if its dynamic priority is higher than the ceiling of any currently locked resource
 - excluding any that it has already locked itself

$$B_i = \max_{k=1}^k usage(k, i)C(k)$$





Blocking

- Rule to calculate “usage”
 - “Longest critical section in any of the lower priority processes that are accessed by higher-priority processes”
 - Or,
 - Resources which have ceiling equal or higher than i,
 - and, are accessed by processes lower than i
 - Actually, same rule as before, but now we can talk about “ceilings”

$$B_i = \max_{k=1}^k usage(k, i)C(k)$$

14

Calculate Blocking

Q used by a, d with ceiling 4
V used by c, d with ceiling 4

Ba=0 ; Bb=4 ; Bc=4 ; Bd=max(4,2)=4

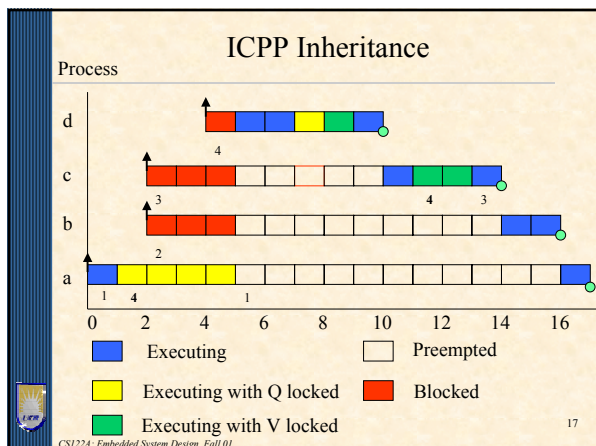
Process	Priority	Execution Sequence	Release Time
a	1	EQQQE	0
b	2	EE	2
c	3	EVVE	2
d	4	EEQVE	4

15

ICPP

- Each process has a static default priority assigned
 - perhaps by the deadline monotonic scheme
- Each resource has a static ceiling value defined
 - this is the maximum priority of the processes that use it
- A process has a dynamic priority that is the maximum of
 - its own static priority and
 - the ceiling values of any resources it has locked
- As a result:
 - Once the process starts actually executing
 - all the resources it needs must be free
 - if they were not, then some process would have an equal or higher priority and the process's execution would be postponed
 - A process will only suffer a block at the very beginning of its execution

16



OCPP versus ICPP

- Although the worst-case scheduling behavior of the two ceiling schemes is identical, there are some points of difference:
 - ICPP is easier to implement than the original (OCPP) as blocking relationships need not be monitored
 - ICPP leads to less context switches as blocking is prior to first execution
 - ICPP requires more priority movements as this happens with all resource usage
 - OCPP changes priority only if an actual block has occurred

18

What have we learned?

- Embedded systems introduction, design metrics, Moore's law
- Custom single-purpose processors: hardware design
- General-purpose processors: software
- Standard single-purpose processors: timers, UARTs, A2D, PWM
- Memories: ROM, EEPROM, FLASH, RAM, NVRAM
- Interfacing: Interrupts, DMA, arbitration
- Design of a digital camera
- Advanced state machines, process networks
- IC Technologies: Full custom, Standard Cell, FPGA
- Synthesis, verification, Intellectual Property, core, methodology
- Reliability and Fault Tolerance: N-version, Recovery block
- Scheduling: CE, FPS, EDF, RMS, DMS, PI, ICPP, OCPP



CS122A: Embedded System Design, Fall 01

19

The End

20

Administrative matter

- Final Examination
 - Tuesday 12/10, 8AM-11AM, here
 - 25% of your grade
 - Material breakdown
 - 10% All Labs
 - 90% ESD chp 8,10,11, RTSPL 5, 13.1-13.11
 - Scope
 - Everything covered in the lecture, the books, or the lab



CS122A: Embedded System Design, Fall 01

21