(2)

logic control - program control flow, i.e. effective seq. of operations to be executed - fundamental to define (WCET)

temporal control - control of the execution instants of the program (activations)

trigger tasks:
- by time, by events (asynchronous, by change of system)
  - well def. WCS
  - poorly defined WCG - using probabilistic arguments

=> CPU utiliz. is constant when there are no deviations in the system

utilization = sum (exec time / activation period), 1..N

temporal complex - measure of how the exec. time grows with problem size O()

task scheduling - seq. of task execution in 1 or more processors, Gantt chart

schedule is feasible - if meets all constraints acc. to the task set

task set is schedulable if there is at least 1 feasible schedule

scheduling problem: IN: task set + constraints
OUT => final assignment of processor time to tasks
- => meet all constraints
- => exec. tasks completely

sched. alg. - preemptive × non-pree
static × dynamic
offline × online (sched. defined with the / before system execution)

EDD: $J = \{J_i (C_i, a_i = 0, D_i) \; i = 1..n\}$
increasing deadline order

EDF: $J = \{J_i (C_i, a_i, D_i) \; i = 1..n\}$, preemp.
earliest deadline minimalizes max lateness
$L_{max}(J) = max_i (f_i - d_i)$, $O(n \cdot log(n))$

BB - branch and bound, non-preemptive
- all possible permutation ($O(n!)$)

static cyclic scheduling
- MC
- microcycles with fixed durations
- => determine the cycles in which tasks are activated

(+) simple, low exec. overhead, allows optimizing
(−) not scalable (small change -> cause changes in table)

online sched. with fixed priorities
- => schedule is built while the system working
- => ready queue is sorted by descending priorities

(+) easily scalable, easily accomodate sporadic tasks, deterministic behavior over overloads
(−) more complex implementation, higher run overhead, high prior tasks can stuck!

schedulability tests
- based on CPU utilization
- based on response time

(RM) $U(n) = \sum_1^n \frac{C_i}{T_i} \leq n(2^{\frac{1}{n}} - 1)$

$U(n) > 1$ -> non schedul.
$U(n) \leq LUB$ => schedulable
$1 \geq U(n) \geq LUB$ -> indeterminated case

Deadline-Monotonic - RTime
$U(n) = \sum \frac{C_i}{D_i} \leq n(2^{\frac{1}{n}} - 1)$

WCRT: $Rwc_i = max(f_{ik} - a_{ik})$

$Rwc_i \leq D_i$

$Rwc_i = I_i + C_i$

interference $I_i = \sum_{k \in hp(i)} \lceil \frac{Rwc_i}{T_k} \rceil \cdot C_k$

$Rwc_3(0) = C_1 + C_2 + C_3 = 4$

$Rwc_3(1) = \lceil Rwc_3(0)/T_1 \rceil \cdot C_1 + \lceil Rwc_3(0)/T_2 \rceil \cdot C_2 + C_3 = 5$

Dynamic priorities (Online) known at run time

(−) instability with overloads - impossible to know a priori which tasks will meet their deadlines
higher run time overhead => Bad Slackford
assign. priorities: EDF, LSF, FCFS => no preempt.
more fair

perf: CPU utilization, respond time, CPU demand

$D = T \Rightarrow U = \sum_{i=1}^n \frac{C_i}{T_i} \leq 1 \Leftrightarrow$ sched. set
$D < T \Rightarrow U^* = \sum \frac{C_i}{D_i} \leq 1 \Rightarrow$ —||—

EDF: 1 instance per period assured, not with RM

fairness - lastly meeting deadlines; avoids preemptions
use a stack of tasks of faster activation / larger jitter in the prior. tasks

L - synchronous busy period (longest interval of consecutive CPU load (activity))
$L(0) = \sum_i C_i$
$L(m+1) = \sum_i \lceil \frac{L(m)}{T_i} \rceil \cdot C_i$

demand analysis
$h(t) \leq t \;\; \forall t \in [0, L]$ | $S = \bigcup S_i$; $S_i = \{m \cdot T_i + D_i, \; m = 0, 1..\}$

RT is more complex, can be: $\forall_i \; Rwc_i \leq T_i + U$ (pessimistic)

LSF - priority of ready tasks increases as time passes, P of exec. remains, higher number of preemptions

FCFS - good fairness, P age of tasks -> ↑ prior, arriving task -> lower prior

shared resources: priority inversion - task that blocks lower priority
idle -> R <- B, (E)
-> task execute for a longer time (C+B) } blocking, ONCE!

synchro prim: interrupt disabling (ID), preemption dis, locks, semaphores

ID - all activities are blocked,

semaphores - affect only tasks in sharing resource
avoid: chain blocking, undetermined blocking, deadlock

PIP - blocking task inherits the priority of blocked task (HP)

PIP < indirect blocking (of intermed. tasks), direct blocking of H

(−) chained B, not DL free

PCP - control free semaphor
task acquire sem if it's free and its prior is higher than the ceilings of all semaphor

SRP - task can start execution only if its preemption level is higher than that of the exec. tasks and higher than ceilings of all curr. locked semaphores

(+) no blocking, no need of indiv. stack

① Spectrum

FPGA ⇒ microcontr ⇒ microproc. ⇒ system on chip ⇒ SBC

other: number of I/O, proc. capacity, power consumption, size

F ⇒ ID ⇒ EX ⇒ MA ⇒ WB

limitat: non-unif. delay, branching

enhanc: out-of order execution

RISC - smaller no. instruction
reduce transistor count
reg: more work out of PC

A prof (App)
R (RT)
M (Microcon.)

Memory - RAM - DRAM, SRAM (faster retrieval)
- hybrid - Flash, EEPROM (electr. erase; programmed during manuf.)
- ROM - PROM, EPROM, Masked ROM
EPROM - once programmed
- erased by UV light

Buses

I2C
SDA (data line)
SCL (clock)

transmitter, receiver

master - init transfer, gen. clocks, terminates -||-

slave - addressed

- supp. MULTI-MASTER

collisions - procedure to ensure that if more than 1 M simult. tries to control the bus, only 1 allowed, occurs on SDA line; each master that attemps to transmit 1 but reads 0 ⇒ aborts F transmission

SCL at 1 ⇒ stable data
0 ⇒ changing -||-

SDA never changes state when SCL is at 1 except
1.) marks the beginning of a F
2.) STOP

• always 8 bit seq., in the end ⇒ ACK bit 0 by receiver, low 1.5V / high 3V

S	slave addr	R/W	A	data	… data	A/A	P

Broadcast [0000000] A [xxxxxxx B]
B=1 used by master, sec. identif device
B=0 sec byty 0x06 reset addr + res device / 0x04 res addr

standard - 0-100kBit/s
fast - 0-400kBit/s
H.S. - 3,4Mbit/s

• no timeout

SMBus
max speed: 100kHz
min: 10kHz (slow!)
timeout 35ms
low 0.8V, high 2.1V
allows slaves to stretch each bit, as long as min speed of 10kHz is met

SPI - full duplex, serial per. interf.
MOSI, MISO
SS - slave sel.
SPI data reg.
not stand. timing
proces. periph.

USB - universal serial bus
1 master, n slaves
master init. all data transm.
M period. polls each slot
VBUS, D-, D+, GND
comm. betw. host and a device over virtual pipes
each pipe ⇒ end-point in the device
each device ⇒ function

Cross compilation
prog.c ⇒ compile ⇒ prog ⇒ exec.
cross-comp ⇒ prog
host platform ≠ exec.!

how exec. prog?
by interpreting the ...

Validation memory
1) non-par (8 bits)
2) parity (8 bits + 1 par)
3) ECC (8 bits + 1 ECC bit) error code correction

pre-processor gcc -E
compilation gcc -S (prog. into assembly)
assembly gcc -c (.s ⇒ machine code)
linking
exec. prog | gcc -c hello.c

assembly with disassembly
hello.s ⇒ hello.o, not exec.
⇒ contains unresolved symbols
⇒ must be linked (linking relocatable code of several files and libs)

linking - static (fu copied into executable)
- dynamic (executable contains reference to lib fu)
code is read from the library every time program is run (shared libraries, W-DLL)

How to portably compile programs?
⇒ use auxiliary tools (GNU Build Syst)
- configure ⇒ make
⇒ changes Makefile according to the desired architecture

Memory
stack - local v.
heap - DAM with malloc (free(…))
- data seg.
- text segment
local var, heap / glob, exe code

mem. divided partitions
prot. - process ⇒ limited size of prog
virtual memory

every process its own virtual addr. space
MMU: virtual ⇒ physical
1101 … page table (TLB) offset

- process = own unique PT
- less often used mem. blocks are stored on HD

Timing constraints
RT computing - result of computation must be logic. correct & produced on time

$\forall n, s^n_{out} - s^n_{in} < \sigma$
$\forall n, s^{n+1}_{in} - s^n_{in} < T$

• when a computing system is able to keep the pace of given process and, if needed, influence it in desired way ⇒ RTS

objectives of RTS discipline:
- design, analysis, verif

dependence with time, temporal aspects:
• exec. time of its comput., • response time to events
• regularity of generating period. events

exec time - code structure, OS, DMA, caches
respon. time - multi-tasking (MT), access to shared resources (SR), HW interrupts

functional req: • acquiring environment data
• human-machine interface (HMI)
• direct digital control - feedback
- entities accessible by local images, forms RT DB

temporal req: sets constraints to:
observation delays
computing delays of control rules
delay variations of the process / jitter

sensor reading actu.
not in level reading
vari. in value activation delay

CONSTRAINTS (TC)
soft - TC according to which the exec. not satisf. the utility after D
firm - not lose the utility
hard - TC when not met ⇒ catastroph. failure

soft RT - only F & S TC
hard RT - at least 1 H TC, safety critical systems
well defined worst-case scenario
• system must possess resources to withdraw a WCS without need of probabilistic!!

reactive model - program executes in infinite a seq. of interactions IN ⇒ OUT
RT-model = reactive model in which the OUT stream must be synchron. with input stream

determinism - feeding prog with same seq. of inputs ⇒ same outputs
predictability - feeding same IN seq ⇒ same OUT seq. with KNOWN delay or within a known TIME WINDOW

tasks - infin. seq. of activations each exec. sets of functions (jobs)

periodic: $a_n = n \cdot T + \phi$
sporadic - min / min time between activations
aperiodic - stochastically

(C, D)
0, a_0, a_0, s, f_0, a_1
T or min

tasks req: temporal, precedence, resource usage (need for using SV)

preemption - when task can be suspended
interference
blocking

temporal requirements
deadline - upper bound (B) to finish
window - lower and upp. B / instance of T
synchronism - upp. B to the interval between 2 activations
distance - upp. B to the interval betw. the finishing T of 2 consecutive jobs

$\tau_i = \tau_i(C_i, \Phi, T_i, D_i)$ period. | $\tau_i(C_i, mit, D_i)$ sporadic