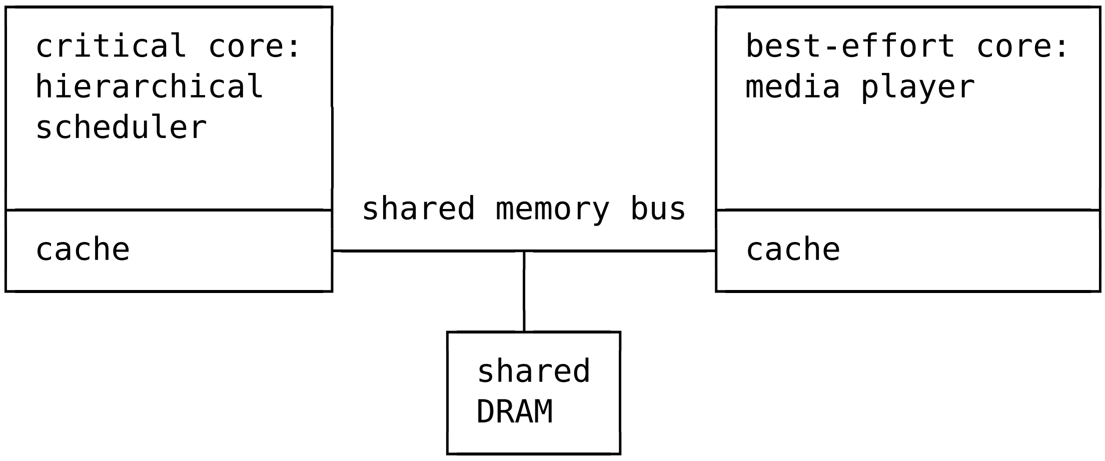
3 system overview: two cores, shared DRAM

The system consists of two cores: the *critical* core, and the *best-effort* core. Those cores have their own caches but share DRAM and memory bus:


critical core: hierarchical scheduler
cache
shared memory bus
best-effort core: media player
cache
shared DRAM

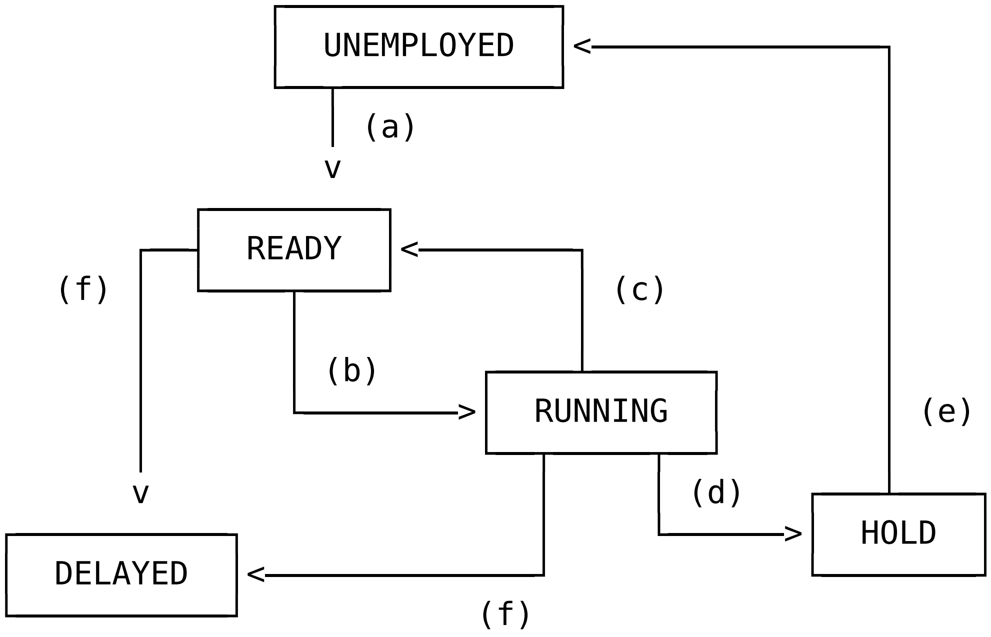
3.0.1 the critical core

The critical core runs critical tasks. The mixed-criticality scheduler[6], as well as all the critical groups, reside on the critical core.

6. Definition of **scheduler**: Computers are capable of executing several programs concurrently – or seemingly so. Actually, the CPU switches from program to program, executing each just so much as to create the illusion that everything happens in parallel. This makes for responsiveness to the user, but also better use of hardware resources: for example, while one program waits for user I/O, it yields the CPU to another, computational-intense program, as the first program cannot make use of the CPU anyway, lacking the I/O data. A special program descides what program executes, and when: the *scheduler*. This program is part of the kernel as it is needs access to the memory. There are several scheduling algorithms that the scheduler can employ to make its choise what program to run. Programs are associated a run-time data strutcture - the process - that will provide the scheduler with uniform metadata on the state of the program, in execution. That data will be used as input, along with the scheduling algorithm, and based on that input, the scheduler will descide what process will get access to the CPU.

5 The task finite state machine

Each individual task belongs to one (and only one) state, always.



5.1 transitions

Transitions occur regularly, and at the same time for all tasks, namely when the global scheduler interrupts execution to reasses the state of the system.

(a) The task is employed; it arrives. This happens after a semi-random delay: in practice, the task should arrive sooner rather than later.

(b) The task has the most immediate deadline in the system of all tasks that belong to non-depleted schedulers; the task is passed to the CPU for execution.

(c) The task is preempted by another task from a non-depleted task scheduler, a task with a more immediate deadline.

(d) The task is completed and is released by the CPU.

6 System model

6.1 the top-scheduler

g is the singular (root) top-scheduler; it schedules task-schedulers:

$$\mathtt{g} = (\mathtt{l}, \mathtt{p}, \mathtt{r}, \mathtt{S}, \mathtt{a}) \tag{1}$$

parameter	name	(unit)	description
l	lifetime	seconds	The system lifetime. This is the amount of time that the top-scheduler will execute, from the time of invocation. As the top-scheduler is the topmost body of the system, l is equal to the system lifetime as well.
p	period	seconds	The global period. At the very beginning of each period (which is equal to the very end of the preceding period), the task-schedulers (that is, all $\mathtt{s} \in \mathtt{S}$) have their budgets resupplied to their individual, pre-assigned levels.
r	rate	seconds	The scheduling rate by which g will schedule S. At every r, execution comes to a halt, and resumes.
S	schedulers		The set of task schedulers that the top-scheduler schedules.
a	algorithm		The scheduling algorithm that g applies to S, in order to select what task to execute (for all $\mathtt{s} \in \mathtt{S}$). The algorithm updates and maintains the statuses of all other tasks and schedulers as well.

For each remaining (available) student, the greedy algorithm would calculate a score based on certain conditions. For example, each students could start with at score of 100. Then,

(1)	the student is assigned the service [at least once]	-7
(2)	(for each such assignment)	-2
(3)	the student is assigned the *minimum* requirement	-20
(4)	(for each assignment above that)	-5
(5)	the student is assigned the *maximum* requirement	*disqualify*
(6)	the student is assigned one more shift today	-10
(7)	for each shift the same *week*, prior to today	-4
(8)	ditto *month*, prior to this week	-2

The greedy algorithm would pick the student with the highest score. After that, everything would happen again, only for the next slot (and so on).

This is an AI, greedy, search algorithm. It is **AI**, since the score system emulates (or, rather, *implements*, with 100% consistency, and 0% flexibility) how a human would think:

(A)	"All students should at least *try* each service, even if there isn't time to acquire any real skill."	(1)[1]
(B)	"Students should get the same quantity of practice."	(2) (4)
(C)	"Students should acquire skills, experience, and confidence."	(3)
(D)	"Students shouldn't do the same thing over and over again, instead, it is preferable they recuperate and/or assimilate what they have learned."	(5)
(E)	"Individual student activity should be distributed, so the student is given time to digest his or her experience, discussing it with fellow students, and so on, to be able to perform better the next time around — also, that will make for a more robust, persistent experience, that he or she won't instantly forget."	(6) (7) (8)

1. This column indicates how a (human) thought has been formalized into a building block of the AI algorithm (as shown in the previous listing). However, as for the AI algorithm's end result, it was my intention that *all* seven point deductions (as well as the one disqualification) more or less should contribute to all five goals.