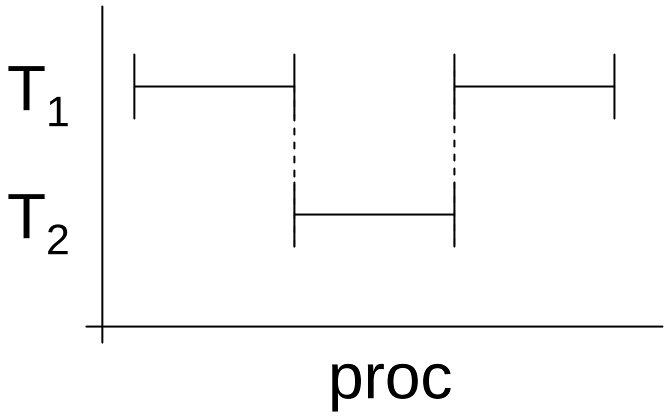
$$\begin{tikzcd}
\text{Enforcing Modularity} \arrow[d] & \\
\text{module per computer} \arrow[d] & \\
\text{module per }\underline{\text{virtual}}\text{ computer} \arrow[d] \arrow[dr] & \\
\text{VM} & \text{Virtual processor}
\end{tikzcd}$$

Virtual Processor

Each program – “Thread” of execution

instructions,
registers, PC, SP
stack



Cooperative vs Preemptive

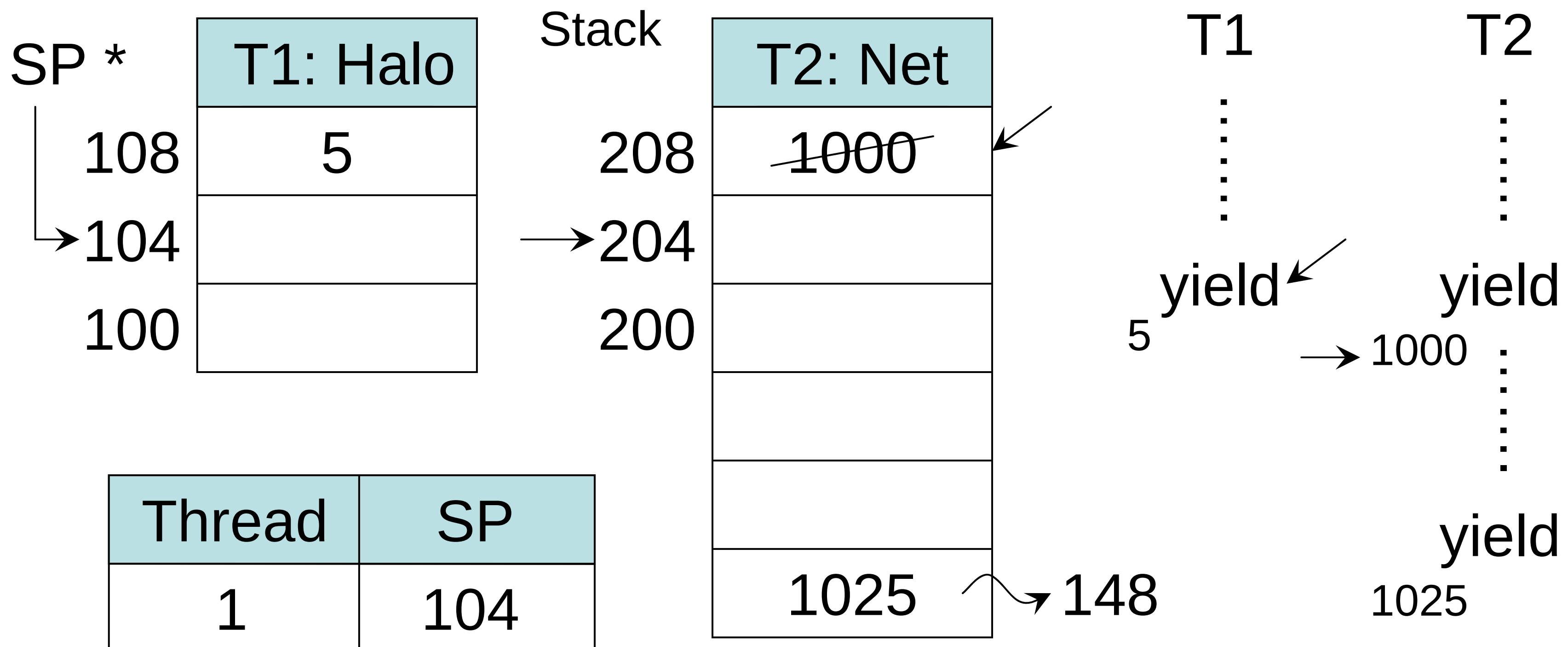
Same AS vs Diff

Yield()

```
yield {
    Save state
    Schedule next thread
    dispatch next thread
}
```

```
int table[NUM_THREADS]
int next
int me ← local to thread
```

Stack Example



Thread	SP
1	104
2	~~204~~
	148

Preemptive scheduling

(No explicit yield)

Timer interrupt $\nearrow$ Processor line

checked by µProc before each instr

If high, calls “gate”

Kernel calls yield() on current thread

Save state

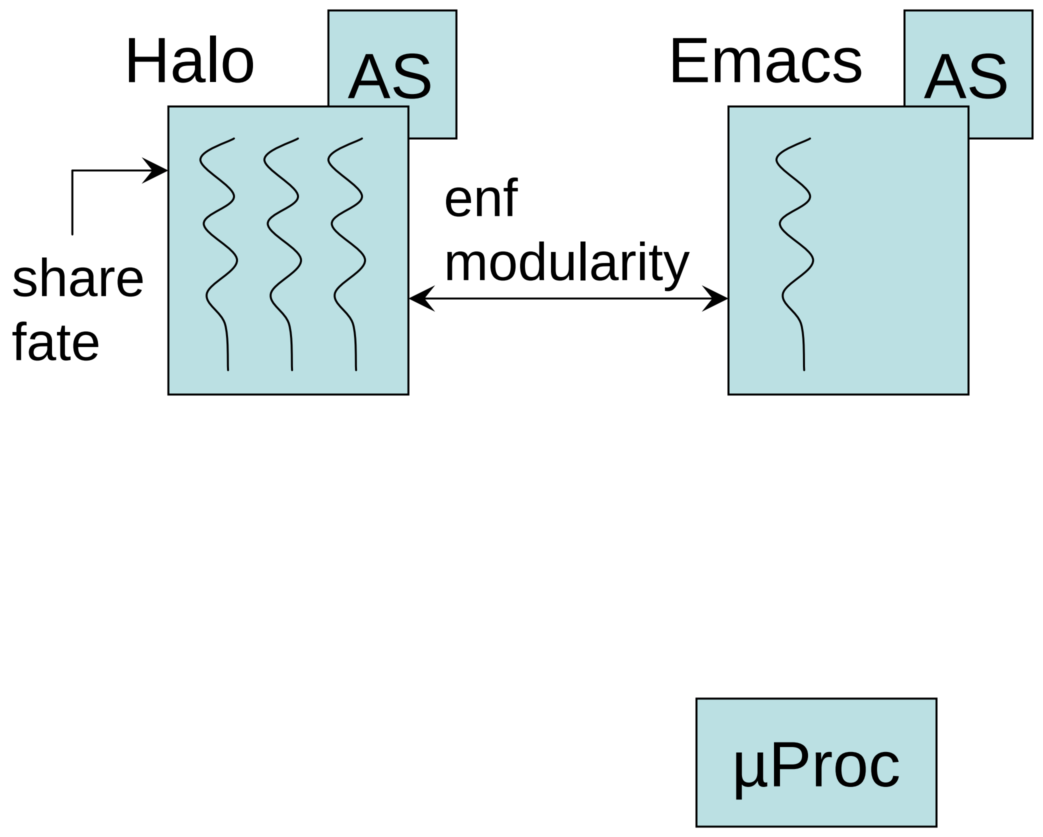
schedule + run next thread

Processes – AS + thread(s)

Kernel support

- create
  - alloc AS, phys. mem, page map
  - load code into mem, map into AS
  - create thread, add to thread table
- destroy →
  - remove AS
  - remove thread from table

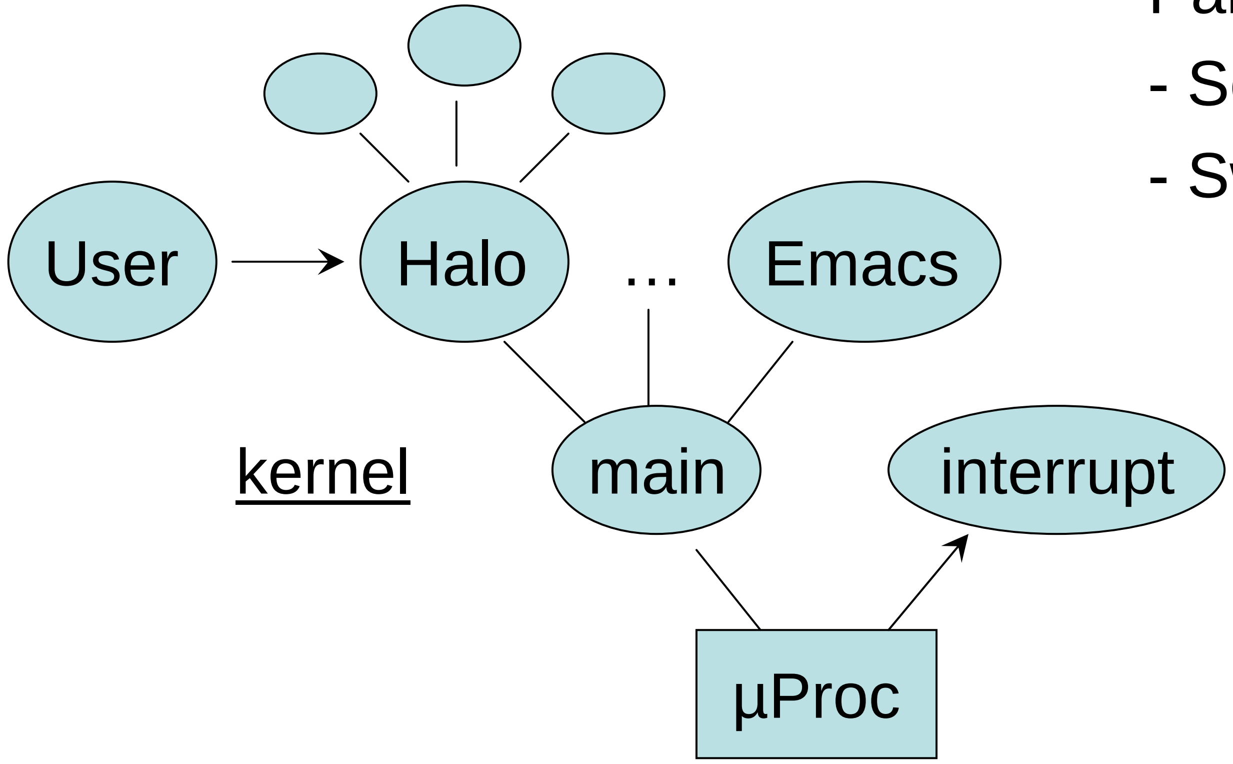
Layering of Threads



Parent threads

- Scheduling policy
- Switching mech.

Layering of Threads

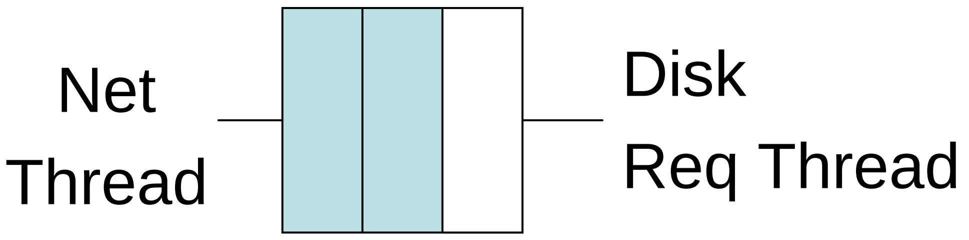


Parent threads

- Scheduling policy
- Switching mech.

Coordinating Access

Web Server



Sequence Coord

```
wait(v, cond)
signal(v)
```

```
while(true)
    while(empty());
    m = dequeue()
    process(m)
```

```
while(true)
    m = next_blk()
    while(full()){};
    enqueue(m)
```