Timer interrupt

+-----------------+
|                 |
|                 |
| t               |
|                 |
|                 |
|                 |

times

the CPU can

intercepts

4
When PS is in stack
Load PC into Instruction Register
Push PS
Push PC
Load PS
Save PC
Push PS
Return
Halt
CPU executes the instruction
Stack of R2

LOAD sp ← pc of R2 (sp points)

sp ← sp + 1

save context

Just Lambda
pop Rx ← copy new constant

push Rx ← copy Rx by constant

Spy ++
big Spy to Rx
from relocation pointer
Can runs P?

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Pc</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
</tr>
</tbody>
</table>

Copy so $\rightarrow P$
Sincerely -

Concerned since - much from myself

Conduct your business well

Start constantly working for/with

Each person has a stage

This is a core (be)

Lots of processes
Some kind of process flow:

\[ \text{process} \leftarrow \text{produce} \]

\[ \text{produce} \leftarrow \text{produce + address + space + stack + CPU + registers + PTE} \]

\[ \text{on chip} \]

\[ \text{examine} \}

\[ \text{some kind} \]
Thread → the activity in a process
→ a 'process' can have multiple threads
→ threads are like processes
→ threads share data of a process
a pig / TGD
share with
best do not
and eat
and hear
from dad
share this
grand child
child
father (grand)
parent press
Scheduling queue:

- NEW
- READY
- RUNNING
- WAITING

Diagram showing the transitions between these states.
CPU scheduler

READY

RUN

WAIT