Exceptions

Normal flow control
1. next instruction immediately follows
2. " " result of jump branch cell

Exceptional flow control
1. external event (e.g. interrupt)
2. internal event (e.g. page fault, div by zero)
3. explicit request (e.g. ask for controlled resources)
<table>
<thead>
<tr>
<th>Name</th>
<th>Timing</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 int</td>
<td>async</td>
<td>next inst.</td>
</tr>
<tr>
<td>2 fault</td>
<td>sync</td>
<td>curr inst. (maybe)</td>
</tr>
<tr>
<td>2 abort</td>
<td>sync</td>
<td>never</td>
</tr>
<tr>
<td>3 trap</td>
<td>sync</td>
<td>next inst.</td>
</tr>
</tbody>
</table>
New Process

↓

curr_inst.

↓

next_inst.

Exception

↓

exception handler code

or

exit

OS
Explicit table of pointers to exception handlers

\[
\begin{array}{c}
\text{Code for exc. 0} \\
\text{handler for exc. 1} \\
\end{array}
\]
Interrupts

Action: complete curr inst.
branch to vectored int. service routine (ISR)
return to next inst.

Hints When Writing ISRs

keep it short
give important work for later
Faults

activation branch to except handler
handler decides whether to abort or return to error inst.

Abort
activate terminate process
To ap

Act.

branch

as provider requested service

ref. to next inst.
Processes

process is an instance of a running program

logical control flow

private address space

\[
\begin{align*}
\text{time} & \quad | & \quad \text{PC} & \quad | & \quad \text{PB} & \quad | & \quad \text{PA} \\
\text{time} & \quad | & \quad T & \quad | & \quad T & \quad | & \quad T \\
\end{align*}
\]

concurrency
$T_{m} \mid pA \rightarrow pB \rightarrow pC$

parallelism

Context switch

$\ldots$

Context switch
fork
int fork (void)

if (fork()) == 0)
    printf("child");
else
    printf("parent");
void for(){
    printf("Lo ");
    printf("Ll ");
    printf("Bye");
}

3
void exit (int status)

parent reaps status via wait ()

exit ()