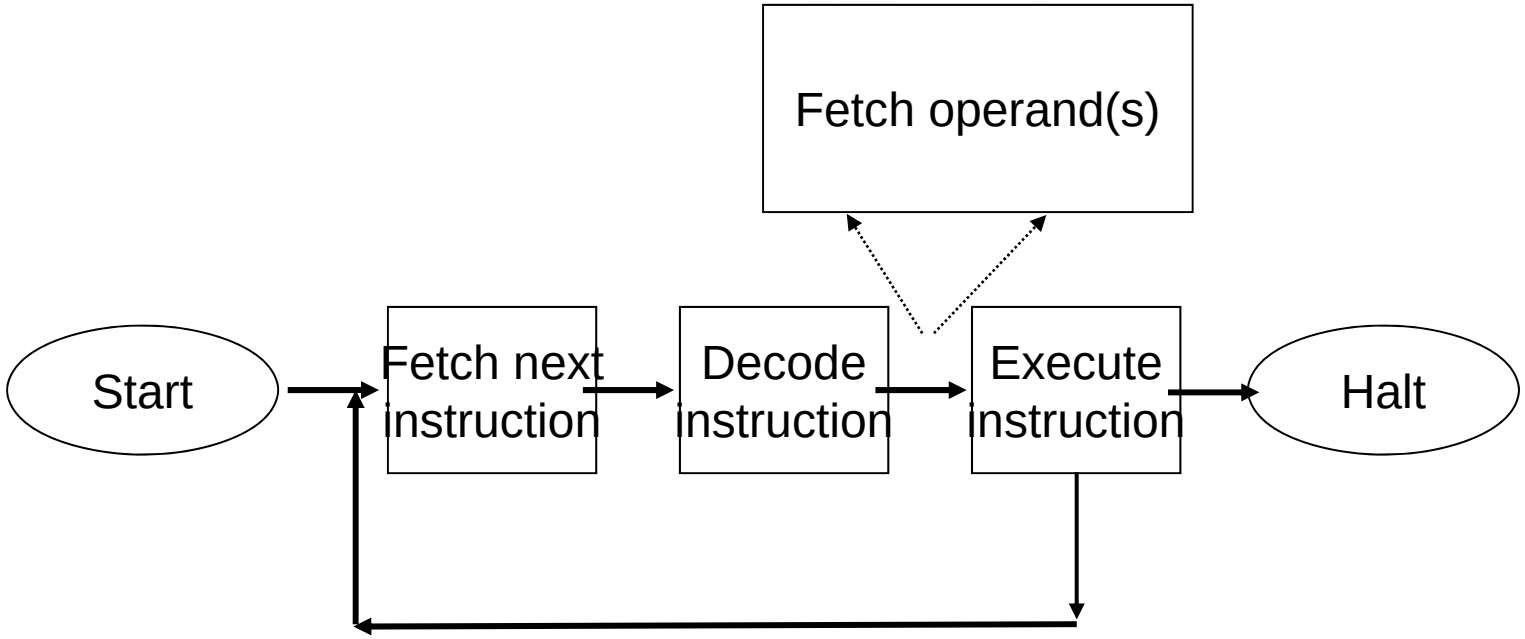
Interrupts

Interrupts

Stefano Quer and Pietro Laface
Dipartimento di Automatica e Informatica
Politecnico di Torino

Instruction cycle

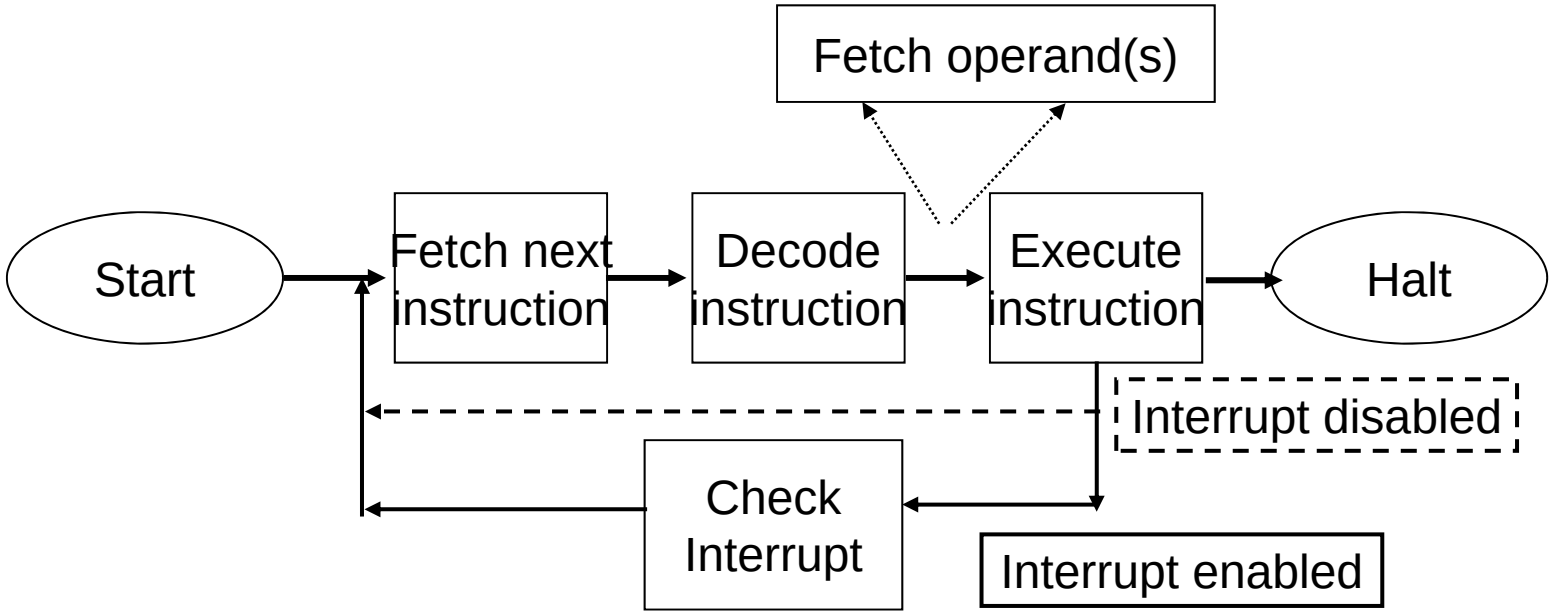


Interrupts

- **Interrupt** is a signal to the CPU generated by hardware or by software indicating an event that needs immediate attention

- **Interrupts** are generated by timers and devices
  - are **asynchronous**, i.e., they are generated at unpredictable times, or during the execution of any program instruction

Instruction cycle with interrupt



Interrupts

- An interrupt signal makes the control flow of a CPU to be moved from the current executing code to an interrupt handler routine that executes another code before returning to the original code.
- It is implemented by
  - saving the current value of the program counter (PC) and status (PSW) registers into a stack, so that the interrupted code can **restart from the next instruction**
  - loading in the PC register the address of the routine corresponding to the specific interrupt

Program Status Word

- The PSW contains
  - condition codes
  - interrupt enable/disable flags
  - kernel/user mode flag
  - ...

Interrupt Vector

Interrupt Handler	116	int_h_10()
	……	
	108	
	164	iret
……. ……. …….		……. ……. …….
main	20000	
	…….	
10	20064	
	20068	
	…….	
	23000	
……. ……. …….		……. ……. …….
Stack	52540	main PSW
	52544	20068
	52548	
	……	

Memory	Address	Content
Interrupt vector	6	
	10	116
	14	PSW of int_h_10()
	16	
	……	……

PC	20068
SP	52548
PSW	main PSW

Issues

- An interrupt needs fast processing, that can be obtained splitting the task in two phases
  - Urgent or critical operations (e.g., get a keyboard code)
  - Operations that can be delayed (e.g., manage the code according to its meaning)
- Nested interrupt processing
- Processing of critical regions with disabled interrupts

Enable/Disable Interrupt (Intel)

- Each interrupt is identified by a number between 0 e 255, which Intel calls **vector**
- The assembler instructions
  - disable interrupt **cli**
  - enable interrupt **sti**

  manage bit **IF** of the register **eflags**, which is tested in AND with masking

Interrupt management

- Disable interrupts while an interrupt is being processed
  - Processor ignores any new interrupt request signals
  - Interrupts remain pending until the processor enables interrupts
  - After interrupt handler routine completes, the processor checks for additional interrupts
- Higher priority interrupts cause lower-priority interrupts to wait.
  - Causes a lower-priority interrupt handler to be interrupted

Exceptions

- **Exception** differ from interrupts because they are **synchronous**
  - Program errors
  - System call (**`int`** or **`sysenter`** instructions)
  - Page faults
  - Fault conditions

Exceptions

- **Exception** are divided in 3 groups depending of the value of register **`eip`**, which is saved into the stack when the CPU raises an exception
  - Faults
    - The fault condition can be corrected and the process **can restart from the same instruction**
  - Traps
    - Used mainly for supporting debug
  - Abort
    - The error condition is such that it is impossible to decide which value **`eip`** should have

Exceptions examples

- **Program Errors**:
  - divisions by zero
  - illegal instruction
  - memory parity error
  - . . .
- **Protection violations**
  - memory violation

Exceptions examples

```
#include <stdio.h>
int i, j, *pk; // global variables initialized to 0
int main(){
  scanf("%d", &i);
  j=2;
  j = j / i;   // possible division by 0 exception
  printf("%d\n", j);
// Correct program
  pk = &i; // pk set to the address of variable i
  scanf("%d", pk);
  printf("i contains: %d %d\n", i, *pk);
// Program generates here a memory violation exception
pk = 0;
scanf("%d", pk);// tries to write where pk points to,
                // a memory location out of user domain
printf("i contains: %d %d\n", i, *pk);
return 0;
}
```

Programmed exceptions

- A programmed exception occurs because a specific instruction is executed
  - `int` or `int3`
  - `into` (check for overflow)
  - `bound` (check on address bound)
- Programmed exceptions, or software interrupts, allow
  - implementing **system calls**
  - signal events to the debugger