Программирование и основы алгоритмизации

Лекция 6

Процессы и прерывания

Программы и процессы

- ПРОЦЕСС -- работающая (выполняемая) программа
- ПРОЦЕСС -- Функция + Локальные данные + Выполнение
- Разница между программой и процессом видна, если рассмотреть выполнение рекурсивной функции
  - Программа одна, а порождает множество процессов, вложенных друг в друга

Грубая модель **асинхронных квазипараллельных процессов** в операционной системе

```
struct message { ... данные ... };    // сообщение
void f1()                             // программа - процесс
{
while (1)                             // цикл работы процесса
{                          // процесс f1, ожидающий сообщения
message *p = get_message(f1);
send_message(p,f3);   // передать сообщение процессу f3
}}
void f2()                             // автономный процесс,
{ while(1) { ... }}        // ни с кем не взаимодействующий
void f3()                             // процесс f3, принимающий сообщение
{
while (1)
   {
   message *p = get_message(f3);
   delete p;
   }}
void f4()                             // процесс f4, создающий и передающий
{                                     // сообщение процессу f1
while (1)
   { ...
   message *p = new message;
   send_message(p,f1);
   }}
```

- Асинхронность обусловлена разными временными параметрами процессов (моменты, когда необходима передача или прием, определяются только природой процесса – алгоритмом программы)
- Квазипараллельность – процессы могут протекать на одном процессоре в режиме разделения времени

- СОСТОЯНИЕ ПРОЦЕССА -- совокупность характеристик процесса, позволяющая в любой момент времени “остановить” а затем в любой другой момент “продолжить” его выполнение без изменения результатов его работы
- ПРЕРЫВАНИЕ -- асинхронная прозрачная процедура (функция), вызываемая по внешнему событию
- Понятие **АСИНХРОННАЯ** процедура означает, что она вызывается не как обычная процедура (функция) в языке программирования, то есть в основной программе (прерываемом процессе) отсутствует вызов этой процедуры в явном виде
- Прерывающая процедура "вклинивается" между любыми последовательными шагами процесса, поэтому она должна обладать **ПРОЗРАЧНОСТЬЮ**, то есть ее выполнение не должно влиять на прерываемый процесс
- ПРЕРЫВАНИЕ -- приоритетный процесс, выполняемый по внешнему событию

Механизм прерываний в архитектуре i8086

- Прерывание в IBM PC представляет собой вызов подпрограммы по длинному (far) адресу с предварительным сохранением в стеке состояния процессора (регистр флагов) и полного текущего адреса **(CS:IP)**
- Вектор прерывания - область памяти из 4-х байтов, содержащая начальный адрес подпрограммы обработки прерывания.
- В терминологии Си **вектор прерывания - это указатель на функцию обработки прерывания**. Первые 1024 байта оперативной памяти содержат 256 векторов прерывания, имеющих номера от 0 до 255.

Механизм прерываний в архитектуре i8086 (2)

- В терминах Си такой массив векторов прерываний определяется как

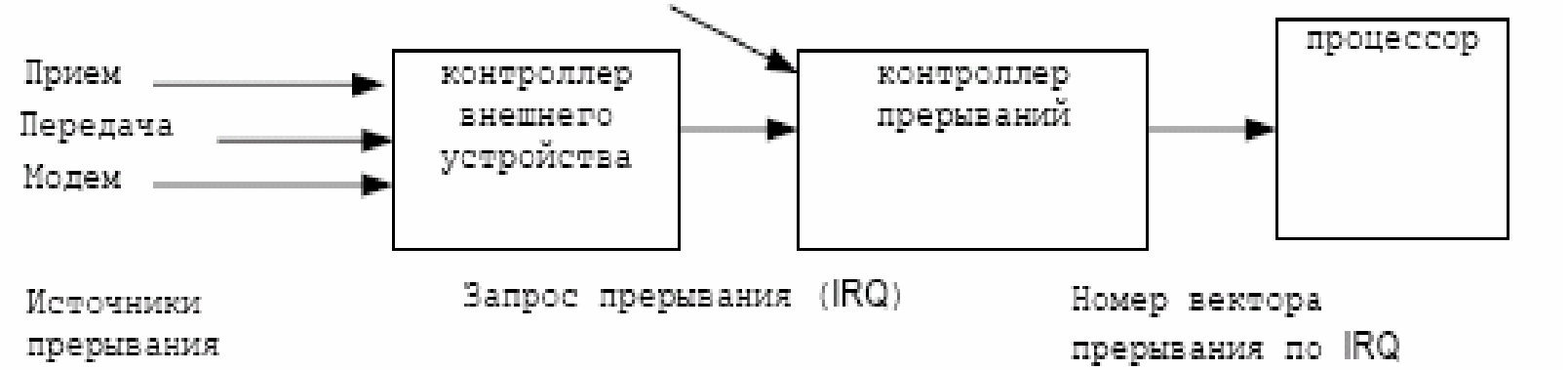
  void interrupt (*IVEC[256])();

- Значение индекса в этом массиве имеет также название - **номер вектора прерывания**
- Каждому источнику аппаратного прерывания соответствует свой номер вектора прерывания.
- Команда **"INT nn"** при выполнении процессором вызывает программное прерывание с номером nn

Механизм прерываний в архитектуре i8086 (3)

- Само вхождение в процесс обработки прерывания, а также управление источниками прерывания осуществляется на нескольких уровнях:
  - на первом уровне устройство, имеющее несколько источников прерывания, при помощи собственных уникальных команд устанавливает “разрешение” или “запрещение” прерывания от каждого из источников. Сформированный таким образом запрос на прерывание поступает на контроллер прерываний по одной из линий запроса. Каждой такой линии присвоен аппаратно определенный приоритет, согласно которым контроллер “пропускает через себя” полученные запросы;
  - контроллер прерываний аналогично имеет возможность “разрешить” или “запретить” прерывание по каждой из линий запроса. Сформированный общий запрос на прерывание с выделенным источником максимального приоритета поступает в процессор;
  - процессор также имеет возможность устанавливать собственное “разрешение” и “запрещение” прерывания, которое представлено соответствующим битом в слове состояния процессора. В Си для этой цели существуют стандартные функции **enable() и disable().**

Механизм прерываний в архитектуре i8086 (4)



Особенности прерывающей программы

- обработчик прерывания должен представлять собой функцию;
- вызов этой функции должен осуществляться асинхронно, то есть ее имя (или указатель на нее) должно быть связано с вектором прерывания процессора;
- функция должна быть “прозрачной” по отношению к исполнительной системе компилятора, то есть выполняемая Си-программа не должна замечать процесса прерывания;
- функция должна иметь доступ как к глобальным (внешним), так и к локальным (автоматическим) переменным

Проблема синхронизации процессов, использующих общие ресурсы

- Если ресурс – время процессора, это задача распределения времени центрального процессора, решаемая операционной системой
- Если ресурс – память или внешнее устройство, это более универсальная задача синхронизации процессов
- Проблема заключается в том, чтобы асинхронные процессы не мешали друг другу через общий ресурс

Проблема синхронизации процессов, использующих общие ресурсы (2)

- Критический интервал – часть процесса, работающая с общим ресурсом, которую нельзя прерывать другими процессами, работающими с этим же ресурсом
- При вхождении в критический интервал необходимо блокировать доступ к этому ресурсу других процессов
- Это делается с помощью операций с переменными-*семафорами* или с переносом критического интервала в специальную программу, время которой разделяется между процессами

Проблема синхронизации процессов, использующих общие ресурсы (3)

- Если ресурс занят при попытке вхождения в критический интервал, процесс должен переходить в состояние ожидания
- При динамическом распределении ресурса между процессами возможна т.н. *тупиковая ситуация* или *дэдлок,* при котором процесс А занял ресурс X и ждет освобождения ресурса Y, процесс В занял ресурс Y и ждет освобождения ресурса X