**БИЛЕТ 22**

1. **Класифікація граматик за Хомським**

**. По виду правил выделяют несколько классов грамматик**. В соответствии с классификацией Хомского грамматика **G** называется:

* **праволинейной**, если каждое правило из **Р** имеет вид: **AxB** или **Ax**, где **A**, **B** - нетерминалы, **x** - цепочка, состоящая из терминалов;
* **контекстно-свободной** (КС) или **бесконтекстной**, если каждое правило из **Р** имеет вид: A****, где A **** N, а **** ****(N **** T)\*, то есть является цепочкой, состоящей из множества терминалов и нетерминалов, возможно пустой;
* **контекстно-зависимой** или **неукорачивающей**, если каждое правило из **P** имеет вид: **  **, где **** То есть, вновь порождаемые цепочки не могут быть короче, чем исходные, а, значит, и пустыми (другие ограничения отсутствуют);
* **грамматикой свободного вида**, если в ней отсутствуют выше упомянутые ограничения.

**Пример праволинейной грамматики**: G2 = ({S,}, {0,1}, P, S), где P:

1) S **** 0S; 2) S **** 1S; 3) S ****, определяет язык {0, 1}\*.

Пример КС-грамматики: G3 = ({E, T, F}, {a, +, \*, (,)}, P, E) где P:

1) E ****T; 2) E **** E + T; 3) T **** F; 4) T **** T \* F; 5) F **** (E); 6) F **** a.

Данная грамматика порождает простейшие арифметические выражения.

**Пример КЗ-грамматики**: G4 = ({B, C, S}, {a, b, c}, P, S) где P:

1) S **** aSBC; 2) S **** abc; 3) CB **** BC; 4) bB **** bb; 5) bC **** bc; 6) cC **** сc, порождает язык { a **n** b **n** c **n** }, n **≥** 1.

Примечание 1. Согласно определению каждая праволинейная грамматика является контекстно- свободной.

Примечание 2. По определению КЗ-грамматика не допускает правил: А ** ** где **** - пустая цепочка. Т.е. КС-грамматика с пустыми цепочками в правой части правил не является контекстно-зависимой. Наличие пустых цепочек ведет к грамматике без ограничений. Соглашение. Если язык L порождается грамматикой типа G, то L называется языком типа G. Пример: L(G3) - КС язык типа G3. Наиболее широкое применение при разработке трансляторов нашли КС-грамматики и порождаемые ими КС языки.

**2.Особливості математичних операцій ММХ-розширення**

**Розширення архітектури ММХ -** основа аппаратной компоненты расширения mmx – восемь новых регистров mm0..mm7, которые на самом деле являются регистрами сопроцессора, только вместо 80-ти разрядов используется 64 младших разряда (мантисса). При работе со стеком сопроцессора в режиме mmx он рассматривается как обычный массив регистров с произвольным доступом. Нельзя одновременно пользоваться командами для работы с числами с плавающей запятой и командами ММХ, а если это необходимо — следует пользоваться командами FSAVE/FRSTОR, каждый раз перед переходом от использования FPU к ММХ и обратно (эти команды сохраняют состояние регистров ММХ точно так же, как и FPU). Основным принципом работы **команд mmx** является одновременна обработка нескольких единиц однотипных данных одной командой.

**Из инета**

Для быстрого снабжения конвейеров командами и данными из памяти шина данных этих процессоров сделана 64-разрядной, из-за чего их первое время иногда ошибочно называли 64-разрядными процессорами. «На закате» этого поколения появилось расширение ММХ (Matrics Math Extensions {instruction set} – набор команд для расширения матричных математических операций (первоначально Multimedia Extension {instruction set} – набор команд для мультимедиа-расширения)). Традиционные 32-разрядные процессоры способны выполнять сложение двух 8-разрядных чисел, размещая каждое из них в младших разрядах 32-разрядных регистров. При этом 24 старших разряда регистров не употребляются, и потому, например, при одной операции сложения ADD осуществляется просто сложение двух 8-разрядных чисел. Команды ММХ оперируют сразу с 64 разрядами, где могут храниться восемь 8-разрядных чисел, причем имеется возможность выполнить их сложение с другими 8-разрядными числами в процессе одной операции ADD. Регистры ММХ могут употребляться также для одновременного сложения четырех 16-разрядных слов или двух 32-разряных длинных слов. Такой принцип получил название SIMD (Single Instruction/Multiple Data - «один поток команд/много потоков данных») (2-4). Новые команды были предназначены в первую очередь для ускорения выполнения мультимедиа программ, но применять их можно не только к задачам, прямо связанным с технологией мультимедиа. В ММХ появился и новый тип арифметики - с насыщением: если результат операции не помещается в разрядной сетке, то переполнения (или «антипереполнения») не происходит, а устанавливается максимально (или минимально) возможное значение числа.

|  |
| --- |
| PADDB приемник,источник – сложение байт PADDW приемник,источник – сложение слов PADDD приемник,источник – сложение двойных слов |

Команды выполняют сложение отдельных элементов данных (байт — для PADDB, слов — для PADDW, двойных слов — для PADDD) источника (регистр ММХ или переменная) и соответствующих элементов приемника (регистр ММХ). Если при сложении возникает перенос, он не влияет ни на следующие элементы, ни на флаг переноса, а просто игнорируется (так что, например, для PADDB 255 + 1 = 0, если это числа без знака, или -128 + -1 = +127, если со знаком).

|  |
| --- |
| PADDSB приемник,источник – сложение байт с насыщением PADDSW приемник,источник – сложение слов с насыщением |

Команды выполняют сложение отдельных элементов данных (байт — для PADDSB и слов — для PADDSW) источника (регистр ММХ или переменная) и соответствующих элементов приемника (регистр ММХ). Если результат сложения выходит за пределы байта со знаком для PADDSB (больше +127 или меньше -128) или слова со знаком для PADDSW (больше +32 767 или меньше -32 768), в качестве результата используется соответствующее максимальное или минимальное число, так что, например, для PADDSB -128 + -1 = -128.

|  |
| --- |
| PADDUSB приемник, источник – беззнаковое сложение байт с насыщением  PADDUSW приемник, источник – беззнаковое сложение слов с насыщением |

Команды выполняют сложение отдельных элементов данных (байт — для PADDUSB и слов — для PADDUSW) источника (регистр ММХ или переменная) и соответствующих элементов приемника (регистр ММХ). Если результат сложения выходит за пределы байта без знака для PADDUSB (больше 255 или меньше 0) или слова без знака для PADDUSW (больше 65 535 или меньше 0), в качестве результата используется соответствующее максимальное или минимальное число, так что, например, для PADDUSB 255 + 1 = 255.

|  |
| --- |
| PSUBB приемник, источник – вычитанние байт  PSUBW приемник, источник – вычитанние слов PSUBD приемник, источник – вычитанние двойных слов |

Команды выполняют вычитание отдельных элементов данных (байт — для PSUBB, слов — для PSUBW, двойных слов — для PSUBD) источника (регистр ММХ или переменная) и соответствующих элементов приемника (регистр ММХ). Если при вычитании возникает заем, он игнорируется (так что, например, для PSUBB -128 - 1 = +127 — для чисел со знаком или 0 - 1 = 255 — для чисел без знака).

|  |
| --- |
| PSUBSB приемник, источник – вычитание байт с насыщением PSUBSW приемник, источник – вычитание слов с насыщением |

Команды выполняют вычитание отдельных элементов даннях (байт — для PSUBSB и слов — для PSUBSW) источника (регистр ММХ или переменная) и соответствующих элементов приемника (регистр ММХ). Если результат вычитания выходит за пределы байта или слова со знаком, в качестве результата используется соответствующее максимальное или минимальное число, так что, например, для PSUBSB -128 - 1 = -128.

|  |
| --- |
| PSUBUSB приемник, источник – беззнаковое вычитание байт с насыщением PSUBUSW приемник, источник – беззнаковое вычитание слов с насыщением |

Команды выполняют вычитание отдельных элементов даннях (байт — для PSUBUSB и слов — для PSUBUSW) источника (регистр ММХ или переменная) и соответствующих элементов приемника (регистр ММХ). Если результат вычитания выходит за пределы байта или слова без знака, в качестве результата используется соответствующее максимальное или минимальное число, так что, например, для PSUBUSB 0 - 1 = 0.

|  |
| --- |
| PMULHW приемник, источник – старшее умножение |

Команда умножает каждое из четырех слов со знаком из источника (регистр ММХ или переменная) на соответствующее слово со знаком из приемника (регистр ММХ). Старшее слово каждого из результатов записывается в соответствующую позицию приемника.

|  |
| --- |
| PMULLW приемник, источник – младшее умножение |

Умножает каждое из четырех слов со знаком из источника (регистр ММХ или переменная) на соответствующее слово со знаком из приемника (регистр ММХ). Младшее слово каждого из результатов записывается в соответствующую позицию приемника.

|  |
| --- |
| PMADDWD приемник, источник – умножение и сложение |

Умножает каждое из четырех слов со знаком из источника (регистр ММХ или переменная) на соответствующее слово со знаком из приемника (регистр ММХ). Произведения двух старших пар слов складываются между собой, и их сумма записывается в старшее двойное слово приемника. Сумма произведений двух младших пар слов записывается в младшее двойное слово.

10 ::: Особливості команд логічних операцій та зсувів

|  |
| --- |
| PAND приемник, источник – Логическое И |

Команда выполняет побитовое «логическое И» над источником (регистр ММХ или переменная) и приемником (регистр ММХ) и сохраняет результат в приемнике. Каждый бит результата устанавливается в 1, если соответствующие биты в обоих операндах равны 1, иначе бит сбрасывается в 0.

|  |
| --- |
| PANDN приемник, источник – логическое НЕ-И |

Выполняет побитовое «логическое НЕ» (то есть инверсию бит) над приемником (регистр ММХ) и затем побитовое «логическое И» над приемником и источником (регистр ММХ или переменная). Результат сохраняется в приемнике. Каждый бит результата устанавливается в 1, только если соответствующий бит источника был равен 1, а приемника — 0, иначе бит сбрасывается в 0. Эта логическая операция носит также название «штрих Шеффера».

|  |
| --- |
| POR приемник, источник – логическое ИЛИ |

Выполняет побитовое «логическое ИЛИ» над источником (регистр ММХ или переменная) и приемником (регистр ММХ) и сохраняет результат в приемнике. Каждый бит результата сбрасывается в 0, если соответствующие биты в обоих операндах равны 0, иначе бит устанавливается в 1.

|  |
| --- |
| PXOR приемник, источник – Логическое исключающее ИЛИ |

Выполняет побитовое «логическое исключающее ИЛИ» над источником (регистр ММХ или переменная) и приемником, (регистр ММХ) и сохраняет результат в приемнике. Каждый бит результата устанавливается в 1, если соответствующие биты в обоих операндах равны, иначе бит сбрасывается в 0.

1. **Використання команд введеня-виведення для управління зовнішніми пристроями**

Початковим поштовхом до розробки ОС були проблеми автоматизації завантаження програм та використання узагальнених механізмів введення-виведення. На початковому етапі розроблення ОС найбільш коштовною частиною був ЦП, тому головною вважалася задача ефективного використання процесору Для цього основною проблемою була організація ефективного введення-виведення з одночасною роботою ЦП над розв’язанням інших задач. Для цього необхідно механізм апаратного переривання, який замінював цикл ЦП з очікуванням готовності даних.

**Структура підпрограми драйверів пристроїв включала наступні блоки:**

* Видача команди на пристрій для його підготовки до обміну
* Очікування готовності пристрою до обміну
* Виконання власне обміну
* Видача на пристрій команди для закінчення операції
* Організація обміну драйвера даними з програмою, яка його використовує.

При створенні мікропроцесорів фактично було повторено процес створення програм обміну для зовнішніх пристроїв, але з деякою систематизацією.

Підключення зовнішніх пристроїв до мікропроцесору виконувалось шляхом визначення вихідного командного порту, вхідного порту стану, які мали однакові номер та вхідного і вихідного порту для введення і виведення даних, які мали однакову адресу. Щоб написати узагальнений драйвер введення-виведення треба визначити адреси портів за допомогою

CMPRT EQU 41H

STPRT EQU CMPRT

INPRT EQU 42H

OUTPRT EQU INPRT

Возвращает в ax 1 байт данных введенных с некоторого устройства

drIn Proc

mov al,cmOn

out CMPRT,al

lwr: in al,STPRT

test al,RdyBit

jnz lwr

in al,INPRT

push ax

mov al,cmOff

out CMPRT

pop ax

ret

drIn endp