Конспект лекций по аппаратному и програмному обеспечению микроконтроллеров и микропроцессоров

Власов Е.В. А9-11
Деревянко Д.А. А9-11
Смирнов В.Ю. А9-11

13 января 2011 г.

Оглавление

Глава 1

Введение

В прошлом семестре мы рассматривали только обработку целых чисел. Почти все современные микропроцессоры обрабатывают числа с плавающей точкой.

Глава 2

Обработка чисел с плавающей точкой

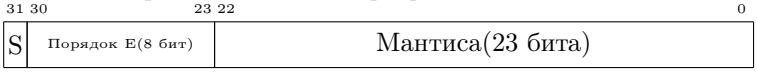
Это будет рассматриваться на примере Pentium.

2.1 Блок FPU (Floating Point Unit)

Прежде чем рассматривать сам блок, мы должны рассмотреть форматы работы с плавающей точкой.

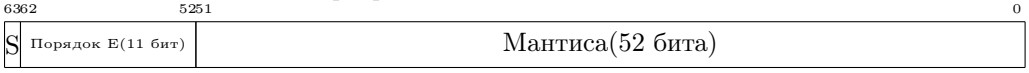
2.1.1 Представление чисел с плавающей точкой

- OT – одинарная точность. 32 разряда.



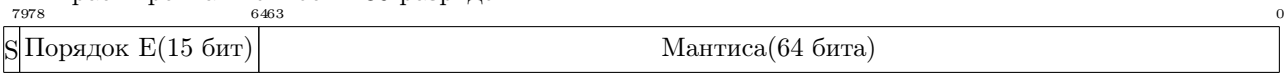
$$\pm 2^{\pm 127} \simeq \pm 10^{\pm 38}$$

- DT – двойная точность. 64 разряда.



$$\pm 2^{\pm 1023} \simeq \pm 10^{\pm 308}$$

- PT – расширенная точность. 80 разрядов.



$$\pm 2^{\pm 16384} \simeq \pm 10^{\pm 4932}$$

Такая точность нужна для избежания ошибок округления.

Представление имеет свои особенности. В поле порядка указывается не настоящий порядок, а смещенный. Это делается для сравнения чисел.

OT: $(-1)^S \times (1.m_1m_2...m_{23}) \times 2^{E-127}$, $m_1m_2...m_{23}$ – мантиса.

E – смещенный порядок

DT: $(-1)^S \times (1.m_1m_2...m_{52}) \times 2^{E-1023}$

PT: $(-1)^S \times (m_0.m_1...m_{63}) \times 2^{E-16383}$ – здесь не указывается единица, так как 64 разряда это много и можно позволить взять лишний.

Особенности представления чисел:

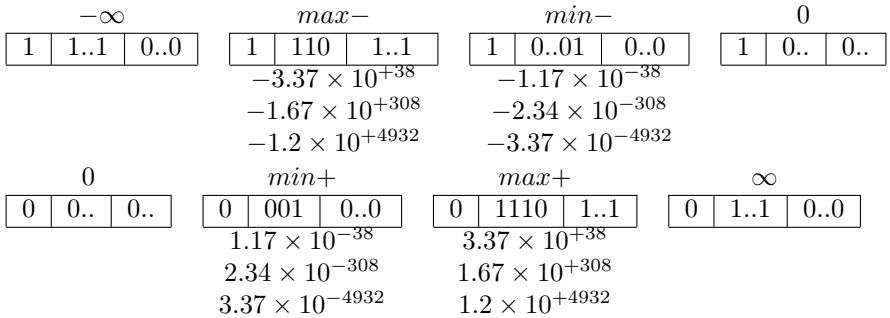
1. Смещенный порядок E.
2. Использование нормализованных чисел.
3. Положительные и отрицательные числа в нормальном коде.

	S	E	M
$+\infty$	0	11...1	0...0
$-\infty$	1	11...1	0...0
$+$	0	00...0	0...0
$-$	1	00...0	0...0
неопр	1	11...1	10...0
NaN	0,1	11...1	$\neq 0$

Не-число NaN – Not a Number. Не подлежит арифметическим операциям.

Неопределенность возникает:

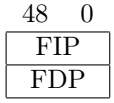
1. $\sqrt{-N}$
2. $\infty \times 0$
3. ∞/∞



Возможно переполнение, когда числа выходят за представляемый диапазон

Регистровая модель

FIP – False Instruction Pointer Offset; FDP – False Data Pointer



- FPSR – регистр состояния.
- FPCR – регистр управления.
- TWR – регитр тегов
- FIP – команда, выз
- FDP – адрес данных (сегмента EA)

Содержимое поля TOP указывает на регистр-вершину. Если, допустим, 011, то третий регистр (P3) является вершиной.

Операции:

1. Вершина стека ST(0)
2. ОЗУ или ST(x)

3. Разрядность стека – TOP в FPSR.

Указатель стека – TOP в FPSR

Теги	
00	число $\neq 0$
01	нуль
10	NaN
11	пусто

TWR	tg7	tg6	...	tg0

2.1.2 Флаги ошибок

Регистр состояния FPSR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B	C3	TOP			C2-C0			ES	SF	PE	UE	OE	ZE	DE	IE

1. IE = 1 – недопустимый результат. Неопределенность. Invalid operation Exception. Происходит при извлечении содержимого пустого регистра или загрузке в заполненный.

   Реакция: прерывание или NaN

2. DE = 1 – денормализованный операнд. Denormalized operand Exception

   Реакция: прерывание или денормализованный результат.

3. ZE = 1 – деление на ноль. Zero divide Exception.

   Реакция: прерывание или результат $\pm\infty$

4. OE = 1 – переполнение. Overflow Exception.

   Реакция: прерывание или результат $\pm\infty$

5. UE = 1 – антипереполнение. Underflow Exception. (Судя по всему слишком маленький результат)

6. PE = 1 – Признак неточного результата. Precision Exception.

   Реакция: прерывание или округление

7. SF = 1 – переполнение стека. Stack Fault.

   Реакция: прерывание

8. ES = 1 – флаг общей ошибки (если есть хоть одна ошибка). Exception summary Status

   Реакция: прерывание

9. TOP – указатель стека.

10. C0-C3 – результат операции сравнения. Condition bit 0-3.

11. B = 1 – операция FPU. FPU Busy.

Регистр FPCR:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				RC		PC				PM	UM	OM	ZM	DM	IM

Маска прерываний

- IM – при NaN или неопределенности. (Invalid operation Exception)
- DM – денормализованный результат. (Denormalized operand exception)
- ZM – ∞ (Zero divide exception enable)
- OM – ∞ (Overflow exception)
- UM – 0 (Underflow exception)
- PM – при округлении (Precision Exception)

Округление

PC	форматы
00	OT
01	-
10	DT
11	PT

RC	округление
00	к ближайшему числу
01	к $-\infty$
10	к $+\infty$
11	к 0

2.1.3 Регистр тегов TWR

Контроль за регистрами R0-R7 (например для определения возможности записи)

$_{15}$ $_{14}$	$_{13}$ $_{12}$	$_{11}$ $_{10}$	$_{9}$ $_{8}$	$_{7}$ $_{6}$	$_{5}$ $_{4}$	$_{3}$ $_{2}$	$_{1}$ $_{0}$
tg7	tg6	tg5	tg4	tg3	tg2	tg1	tg0

2.1.4 FIP, FDP

- FIP – ошибочная команда
- FDP – ошибочные данные

Используются для выяснения и ликвидации причины ошибки.

2.1.5 Способы адресации

1. Регистровая – к регистру стека ST(0) по указателю TOP
2. Адресакция к ОЗУ – как для целых чисел DS:[EAX+EBP]

В исходном состоянии стек пустой. Обращение к пустому регистру – прерывание.

2.2 Системы команд FPU

Если не сказано обратного, то любая команда записывающая в ST увеличивает TOP на единицу.

1. Команды пересылки.
   - FLD addr (DS:[EBP+100]), M→ST(0); В зависимости от режима работы, загружается или 32 или 64 или 80 битные числа.
   - FLD ST(i); ST(i)→ST(0); ST – указатель стека. После загрузки он автоматически увеличится.
   - FST addr; ST(0)→M – обратная к FLD
   - FST ST(i); ST(0)→ST(i)
   - FSTP addr; то же что и выше, но с освобождением ST(0), т.е. регистр становится свободным для записи нового числа. Суфикс P – pull, суффикс освобождение стека.
   - FSTP ST(i);
   - FXCH ST(i); ST(0) $\leftrightarrow$ ST(i); Верхушка стека обменивается с указанным регистром.
2. Загрузка констант (все они с 80 битной точностью)
   - FLDZ; 0→ST(0)
   - FLD1; 1→ST(0)
   - FLDPI; π →ST(0)
   - FLDL2T; $\log_2 10 \rightarrow$ ST(0)
   - FLDL2E; $\log_2 e$ →ST(0)
   - FLDLG2; $\log_{10} 2 \rightarrow$ ST(0)
   - FLDLN2; $\ln 2 \rightarrow$ ST(0)
3. Арифметические операции
   (a) Результаты в новой вершине стека.
      i. Сложение

- FADD addr; M+ST(0)→ST(0)
- FADD ST(0), ST(i); ST(0)+ST(i)→ST(0)
- FADD ST(i), ST(0); ST(i)+ST(0)→ST(i)

ii. Вычитание FSUB ...

iii. Умножение FMUL ...

iv. Деление FDIV ...

(b) С освобождением **старой** вершины стека.

i. FADDP; ST(0)+ST(1) → ST(1), затем освободить ST(0)
FADDP ST(i), ST(0); ST(i) + ST(0) → ST(i), затем освободить ST(0)

ii. FSUBP

iii. FMULP

iv. FDIVP

(c) Специфические операции

i. FPREM ST(0), ST(1); Остаток от ST(0)/ST(1) → ST(0)

ii. FSQRT; $\sqrt{ST(0)}$ →ST(0)

iii. FABS; модуль ST(0) → ST(0)

iv. FCHS; -ST(0)→ST(0)

4. Команды сравнения

(a) С использованием FPSR

- FCOM addr; Сравнить M и ST(0);
- FCOM ST(i); Сравнить ST(i) и ST(0).
- FCOMP – сравнение с освобождением ST(0)
- FCOMPP – сравнение ST(0) с ST(1). Освобождение ST(0) и ST(i)

FPSR

Результат	C3	C2	C0
ST(0)>X	0	0	0
ST(0)<X	0	0	1
ST(0) = X	1	0	0
Несравнимы	1	1	1

(b) С использованием EFLAGS.

- FCOMI ST(0), ST(i); Сравнить с установкой ZF,PF,CF в EFLAGS;
- FCOMIP – сравнение с освобождением ST(0)

EFLAGS

Результат	ZF	PF	CF
ST(0)>X	0	0	0
ST(0)<X	0	0	1
ST(0) = X	1	0	0
Несравнимы	1	1	1

- FUCOMI ST(0),ST(i); сравнение мантисс с установкой EFLAGS.
- FUCOMIP ...; с освобождением ST(0)

(c) FTST; сравнить ST(0) с 0

(d) FXAM; Проверка содержимого ST(0).

Происходит анализ содержимого ST(0) и устанавливается соответствующее значения битов C0-C3 в регистре FPSR. При этом бит C1 принимает значение знака, а разряды C0, C2, C3 определяют содержимое ST(0) в соответствии с таблицей:

	C3	C2	C0
Не поддерживаемый формат	0	0	0
± NAN	0	0	1
норм ± число ≠ 0	0	1	0
±∞	0	1	1
± 0	1	0	0
пусто	1	0	1
± денорм	1	1	0

5. Тригонометрия

   - FSIN; SIN(ST(0))→ST(0)
   - FCOS; COS ST(0)→ST(0)
   - FSINCOS; SIN(ST(0))→ST(0), COS(ST(0)→ST(1)
   - FPTAN; tg(ST(0)) → ST(0)
   - FPATAN; arctg(ST(1)/ST(0)) → ST(0)
   - F2XM1; $2^{ST(0)}$-1 → ST(0)
   - FYL2X; ST(1)$\log_2$(ST(0))→ST(0)
   - FYL2XP1; ST(1)$\log_2$ST(0)+1→ST(0);

6. Функции управления

   - FFREE ST(i); Освобождение ST(0), тэг регистра (tg_i) устанавливается в 11 (признак пустого)
   - FSTSW AX; FPSR → AX (16 разрядов); AX – регистр.
   - FSTSW addr; FPSR→M; addr – 16-и разрядная ячейка памяти, адресованная в команде процессора
   - FLDCW addr; M→FPCR
   - FSAVE addr; FPSR,FPCR, TW, FIP, FDP, ST(10)...ST(0)[94 или 108 байт]→M;
   - FRSTOR addr; M→....
   - FSTENV addr; FPSR,...FDP[14 или 28 байт] → M
   - FDENV addr; M→FPSR...FOP

Глава 3

Групповая обработка данных

В Pentium имеется возможность групповой обработки целых чисел (MMX) и чисел с плавающей точкой (SSE).

Используется принципы SIMD (Single Instruction – Multiple Data).

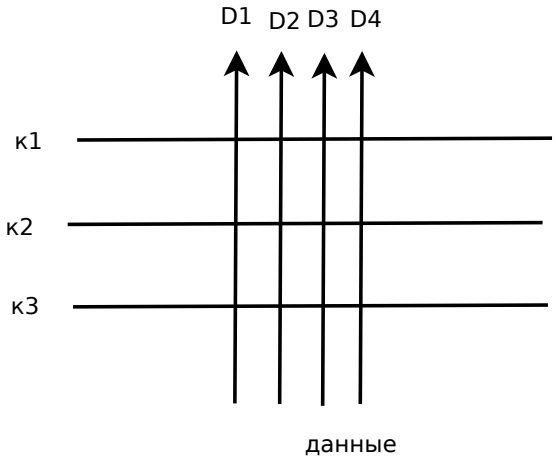


Рис. 3.1: SIMD

3.1 Блок MMX (Multimedia Extension)

Считалось, что они помогут в обработке изображения: сдвиг и другие операции.

Когда его разрабатывали (в 96-97 году) решили сэкономить и поэтому совместили с блоком FPU.

79 64	63 0	теги
	mm0	00
	mm1	00
	mm2	00
	mm3	00
	mm4	00
	mm5	00
	mm6	00
	mm7	00

Работает как обычные регистры.

3.1.1 Формат данных

B7	B6	B5	B4	B3	B2	B1	B0
W3		W2		W1		W0	
D1				D0			
Q0							

B упаковка байтов
W упаковка слова
D – двойного слова (32 бит)
Q – счетверенное слово (64 бит)

Арифметика с насыщением (saturation).

При переполнении результат – максимальное число. Например, при обработке изображений, яркость не может быть больше максимальной.

3.1.2 Способы адресации

1. Регистровая – mmi
2. Обращение к памяти как с целыми числами.

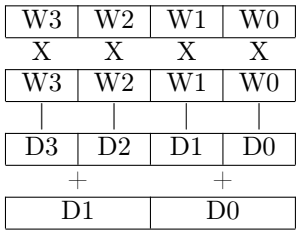
3.1.3 Набор команд

1. Пересылка данных
   - MOVD mmj, mmi/addr; пересылка младшие 32 разряда из mmi/addr в mmj
   - MOVD mmi/addr, mmj; пересылка младшие 32 разряда из mmj в mmi/addr
   - MOVQ ...; пересылка 64
2. Арифметические операции
   (a) Сложение
      i. PADD(B,W,D) mmi, mmj/addr; P – Packed. В зависимости от суфикса или байты или слова или двойные слова. Результат в mmi. mmi+mmj/M $\rightarrow$ mmi. Со знаком без насыщения.
      ii. PADDS(B,W) – со знаком с насыщением
      iii. PADDUS(B,W) – Без знака, с насыщением

   (b) Вычетание
   - PSUB(B,W,D); формат аналогичен сложению; со знаком, без насыщения
   - PSUBS(B,W) – со знаком с насыщением
   - PSUBUS(B,W) – без знаком, с насыщением.

   (c) Умножение
   - PMULLW – умножение слов 16 на 16, сохранение младших разрядов
   - PMULHW – умножение слов 16 на 16, сохранение старших разрядов

   (d) Умножение со сложением
   - PMADDWD



Никакие флаги не выстпаляются при переполнении!

3. Логические операции
   (a) PAND mmi, mmj/M ; И
   (b) POR; или
   (c) PXOR; исключающее или
   (d) PANDN mmi, mmj/M; $\overline{mmi}\&mmi/M$)
   (e) Сравнение
   - PCMPEQ(B,W,D); сравнение на равенство. Если равны 1..1, неравны: 0..0.
   - PCMPGT(B,W,D); сравнение на $>$. Если $>$ 1..1, $<$: 0..0.

   (f) Сдвиг (число сдвигов im8, mmi или M)
   - PSRL(W,D,Q); mmi, im8 или mmj или M; второе число - число разрядов на которые двигаем. Сдвиг вправо. Логический.
   - PSLL(W,D,Q); -‴-; влево логический
   - PSRA(W,D,Q); -‴-; вправо арифметический.

Нахождение среднего, min, max

1. PAVG(B,W) mmi,mmj/M
2. PMINUB mmi, mmj/addr; PMINSW mmi, mmj/addr; (Signed word, Unsigned byte)
3. PMAXUB, PMAXSW ...

3.1.4 Упаковка, распаковка данных

1. PACKSSWB mmi,mmj/M

mmi				mmj/M			
w3	w2	w1	w0	w3	w2	w1	w0

В результате (с насыщением!):

mmi							
b7	b6	b5	b4	b3	b2	b1	b0

b7 - ближайшее граничное значение для w3(mmi), b6 ближайшее граничное для w2(mmj) и так далее. То есть если значение какого-либо исходного слова больше +127, то соответствующий байт будет равен 127, если меньше -128, то, соответственно, -128.

2. PUNPACKLWD; Распаковка

mmi			mmj/M		
w3w2	w1	w0	w7w6	w5	w4

mmi			
w1	w5	w2	w4

3.2 Блок SSE

Групповые операции с плавающей точкой. Работает по принципу SIMD (Single Instruction – Multiple Data, одна команда – много данных). Блок SSE содержит набор специализированных регистров и обеспечивает выполнение набора дополнительных команд. Это расширение впервые появилось в процессорах Intel Pentium III и получило название Streaming SIMD Extension (потоковое SIMD-расширение) или, сокращенно, SSE.

3.2.1 Регистровая модель SSE

127 0
xmm0
xmm1
xmm2
xmm3
xmm4
xmm5
xmm6
xmm7

31 0
MXCSR

Восемь 128 разрядных регистров

127			31		0
			S	E	M
F3	F2	F1	F0		

F0-F3 - числа двойной точности. S- знак E - смещение M - мантиса. В один регистр SSE можно упаковать 4 числа OT.

Регистр управления MXCSR (используются 16 из 32 бит):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FZ	RC		PM	UM	OM	ZM	DM	IM	-	PE	UE	OE	ZE	DE	IE

FZ – реакция на антипереполнение.

RC – округление

PM - IM – маски ошибок, как в FPU.

PE - IE – признаки ошибок, как в FPU.

3.2.2 Адресация:

1. Регистровая: xmmi
2. Обращение к M как для целых чисел.

Адрессация соответствует таковой в MMX.

3.2.3 Команды пересылки

Команды имеют две формы:

1. Суфикс PS – пакетная форма. Обрабатывается 4 числа одинарной точности. (packed) 4 числа упаковываются в один xmmi регистр.
2. Суфикс SS – скалярные операции. Обрабатывается только первое слово. (single) Меняется только F0. F3-F1 не изменяются.

1. MOVAPS xmmi, xmmj/M; xmm/M128→xmm; загрузка выровненных данных (по границе 16 байт)
2. MOVAPS M, xmmi
3. MOVUPS ...; загрузка невыровненных данных.
4. MOVLPS ...; m64→xmmi(F1,F0)
5. MOVHPS ...; m64→xmmi(F3,F2)
6. MOVSS ...; m32→xmmi(F0)

3.2.4 Арифметические операции

1. ADDPS xmmi, xmmi/M (F3-F0)
2. ADDSS ... (F0)
3. SUBPS
4. SUBSS
5. MULPS
6. MULSS
7. DIVPS
8. DIVSS
9. SQRTPS xmmi, xmmj/M; $\sqrt{xmmj/M} \to xmmi$
10. SQRTSS ...;
11. RCPPS ...; $\frac{1}{xmmj/M} \to xmmi$
12. RCPSS ...;
13. RSQRTPS ...; $\frac{1}{\sqrt{xmmj/M}} \to xmmi$
14. RSQRTSS ...;

3.2.5 Логические операции

1. ANDPS xmmi, xmmj/M
2. ORPS ...;
3. XORPS ...;
4. ANDNPS ...; $\overline{xmm} \& xmmj/M$

3.2.6 Сравнение

1. CMPccPS; cc - код условия. Выполнение – результат 1..1 xmm; невыполнено 0..0
2. CMPccSS

   Коды условий:

EQ	$=$
NEQ	$\neq$
LT	$<$
LE	$\leq$
NLT	$>$
NLE	$\geq$
UNORD	не сравнимы

   Несравнимы значит, что хотя бы одно число – NaN.
3. COMISS – сравнение F0 с установкой ZF, PF, CF, CF в EFLAGS. Как в MMX.

	ZF	PF	CF
NAN	1	1	1
$>$	0	0	0
$<$	0	0	1
$=$	1	0	0

3.2.7 Команды нахождения min,max

1. MAXPS ...;
2. MAXSS ...;
3. MINPS ...;
4. MINSS ...;

3.2.8 Команды управления

1. FXSAVE addr; все регистры FPU, SSE, MMX (512 байт) в M
2. FSRSTOR addr; загрузка всех регистров FPU, MMX, SSE из M
3. LDMXCSR addr; M→MXCSR
4. STMXCSR addr; MXCSR→M

Глава 4

Структура микропроцессоров Pentium III, 4, Core Duo

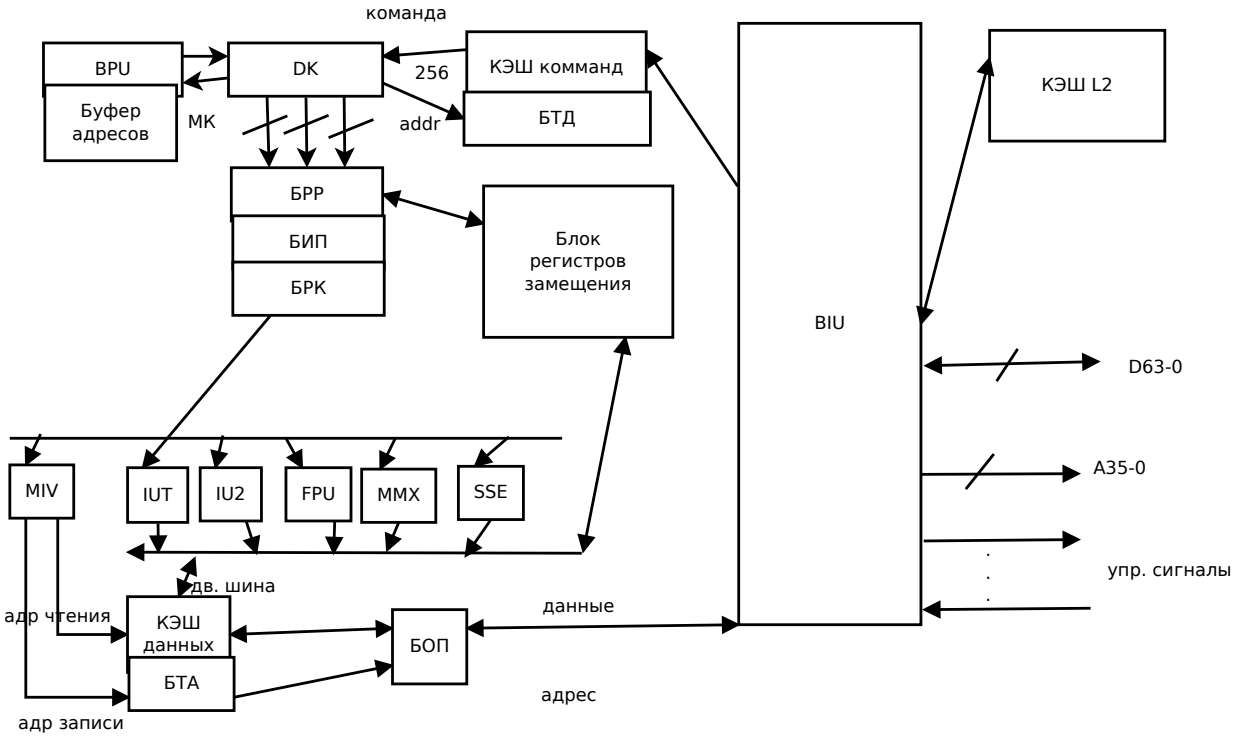


Рис. 4.1: Структурная схема микропроцессора Pentium

- BIU – блок внешних интерфейсов.
- БТА – блок трансляции адресации. Выбор сегмент страницы адресации.
- ДК – дешифратор команд.
- BPV – блок обработки ветвлений.
- MIV – блок формирования адресов.
- IU1,2 – целочисленные блоки.
- FPU, MMX, SSE.
- БРР – блок распределения регистров.
- БИП – блок изменения порядка команд.
- БРК – блок распределения команд.

- БОП – буфер обращения к памяти.

Внутри – гарвардская архитектура. Суперскалярная.
Внешний кэш 2-ого уровня, обычно несколько мегабайт.
Шина адреса 36 бит – позволяет обращаться к 64ГБ памяти.
Из памяти выбирается сразу 256 бит (до 10 команд)
Одновременно три микрокомманды.
В P4 64 регистра.
BPU – предсказатель: дополнительная память – буфер адресов. История ветвлений, 4 бита. Нужна для улучшения предсказаний.
БИП – изменения номера (перестановка) команд.
Уже перестановленные команды поступают в блок распределения.
MIV – формирует 2 адреса – чтения и записи. Одновременно работает до 4-х устройств, в среднем считается что 2.
БОП – блок восстановления порядка

4.1 Эволюция микроархитектуры Pentium III→4→...

Архитектура – IA32
Микроархитектура – внутрение особенности реализации архитектуры.

1. Предварительное декодирование команд в микрокоманды.
   - Сохранение микрокоманд в кэш память
   - Pentium III – кэш команд.
   - Pentium 4 – кэш микрокоманд (12 тысяч микрокоманд, позволяют выполнить 126 команд). В Pentium III - 46 команд
2. Организация гиперконвеера.

   Pentium III - 12 ступеней.

   Pentium 4 – 20 в ранних версиях, 30 в последних.

   Это нужно для увеличения частоты работы, например Pentium III работал на частотах около 1ГГц, а Pentium 4 мог работать на более чем 3ГГц.
3. Усовершенстсования BPU – блока предсказания ветвлений.

   Pentium III – 512 адресов, Pentium 4 - 4096 адресов. В итоге на основании анализа Pentium 4 предсказывает в среднем 9 из 10 переходов.
4. Расширенный набор регистров замещения. Pentium III – 40 регистров замещения (EAX, EDB). Pentium 4 – 128.
5. Дополнительные блоки обработки.

   Pentium III – 2 блока IU, Pentium 4 – 4 блока.
6. Усовершенстсованый блок SSE2.

   Pentium III обработка OT: 128 – 4x32. Pentium 4 – DT, 128 – 2x64
7. Дополнительная обработка целых чисел (дополнение MMX): 128 бит 16x8, 8x16, 4x32.

   Дополнительные 144 комманды групповой обработки
8. Кэш-память 2-ого уровня 256КБайт на кристалле P4.
9. Блоки обработки. Pentium III – 8.5-12 млн. операций. P4 – 42 и в конце более 100.

   Pentium III - 2 команды за такт, Pentium 4 – 3-4 команды за такт.

Глава 5

RISC-процессоры.

Reduced Instruction Set Computer

- 1981 – университет Беркли Стенфорд. Цель процессора – выполнять одну команду за такт.

5.1 Особенности RISC-архитектуры

1. Фиксированный формат команд.

   32 разрядный МП – команды 32 разряда. 8 разрядный - 8 разрядов.

   Уменьшается число тактов простоя и ожидания ⇒ растет производительность. Увеличение эффективности работы конвейера.

2. Исключение сложных способов адресации, таких как: с пост-инкриментом, пре-декриментом.

   Упрощает устройство управления, сокращает объем микропрограммной памяти, позволяет уменьшить размер кристалла и снизить стоимость, позволяет повысить тактовую частоту.

3. Исключение редко используемых команд, а также не вписывающихся в формат

4. Обработка данных с использованием регистровой или непосредственной адресации.

5. Обращение к памяти – команды загрузки и сохранения регистров.

6. Расширенный набор регистров общего назначения. Обычно 32 регистра.

Все это нужно для следующих целей:

- Упрощение УУ.
- Ускорение обработки

5.2 Основные варианты RISC-архитектуры

1. MIPS (MIPS Technology)

   Продажа лицензий на 32 и 64 разрядные чипы. Выпуск примерно 400 млн/год. Используется в принтернах, DVD-плеерах, коммуникационном оборудовании.

   В России выпускаются МП семейства 1890 в НИИ СИ по 0.35мкм тех. процессу (планируется переход на 0.25)

2. SPARC (Sun Microsystems)

   Продажа лицензий. OpenSource. Используется МЦСТ. Технология 0.13мкм.

3. ARM (Advanced RISC Machines)

   Продажа лицензий. В России выпуском занимается НТЦ «Модуль»

4. PowerPC (IBM + Freescale Semiconductors).

   Суперкомпьютеры: x86/x86-64: 70%, PowerPC – 30%

5. SuperH (Hitachi)

5.3 RISC процессоры архитектуры MIPS

1890ВМ2Т (НИИ СИ РАН)

5.3.1 Основные характеристики

1. 3.2 млн. транзисторов
2. 0.35мкм
3. Питание 3.3В
4. Частота 100МГц
5. Мощность 30мВт на МГц.

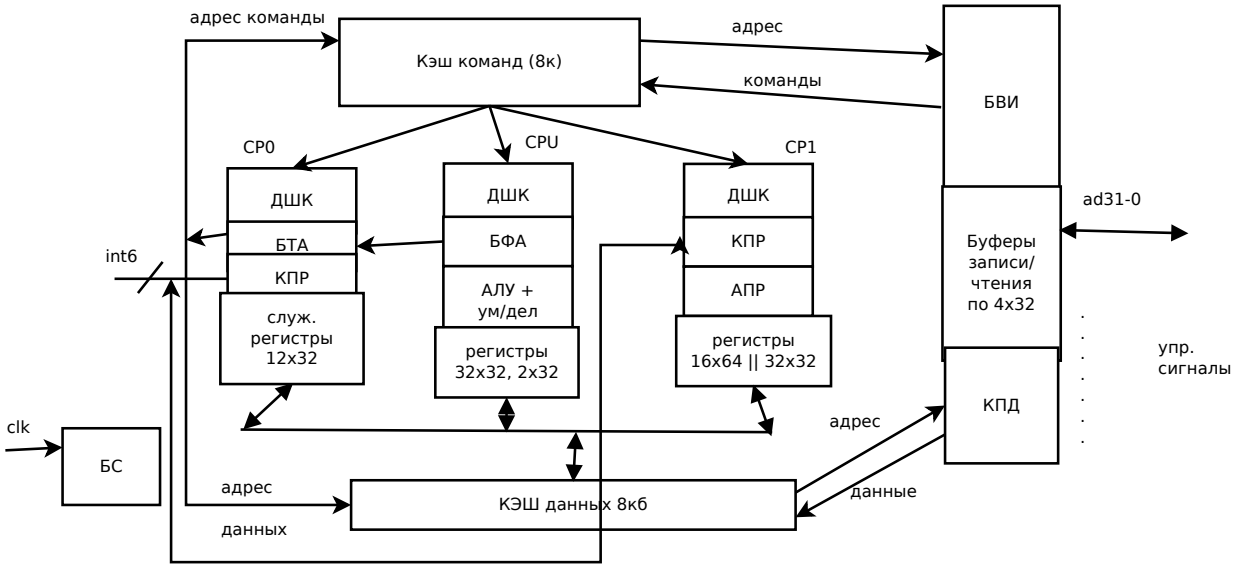


Рис. 5.1: MIPS

- CPU – центральный процессор
- CP0 – сопроцессор
- CP1 – сопроцессор, работающий с плавающей точкой (одинарная или двойная точность)
- БС – блок синхронизации
- ДШК – дешифратор команд
- БФА – блок формирования адреса
- БТА – блок трансляции адреса
- КПР – контроллер прерываний
- АПР – арифметический процессор
- БВИ – блок внешних интерфейсов
- КПД – контроллер прямого доступа к памяти

Буферы чтения и записи 4 слова по 32 байта.
Гарвардская архитектура: отдельные шины адреса и данных. Суперскалярная архитектура: 3 основных устройства (CP0, CPU, CP1), используется страничная адресация.

5.3.2 5-ступенчатый конвейер

1. Выборка и дешифрация команд (IF)
2. Чтение операндов. (RD)
3. Арифметико-логические операции. (ALU)
4. Загрузка или сохранение в ОЗУ (MEM)
5. Запись результатов в регистр. (WR)

5.3.3 Способы адресации

Команды обработки:

- Регистровая (3 адреса)
- Непосредственная
- Косвенно-регистровая со смещением. offset (r_b) Адрес = r_b + offset; Регистры у нас 32 разрядные, а смещения - 16 разрядов со знаком.

  Например Обращение к памяти. LOAD (M→Reg), STORE (Reg→M)

  Варианты:

  1. Положим смещение равным нулю – получим косвенно-регистровую адрессацию без смещения.
  2. r_b=0 → r0=0, тогда адресс = offset. Получается возможность адресовать напрямую первые 16 бит (64КБ).

     К первым 64 КБ можно обращаться через прямую адрессацию, к остальным **только** косвенно-регистровой со смещением.

Вся подобная архитектура нужна для того, чтобы на каждом такте выдавался результат одной операции. Но, как известно, идеал не достижим, поэтому существует целый ряд исключений.

Исключения из правила: 1 операция = 1 такт

1. Остановка конвейера при операциях MUL, DIV. Обычно 20-30 тактов.
2. Задержки после загрузки данных (LOAD). 1 такт. Т.е. код выглядит примерно так:

```
LOAD 2d, addr
NOP
ADD
```

3. Задержка при обработке команд перехода. 1 такт. Конвейер не успевает обработать адрес, поэтому нужно делать так:

```
JUMP addr
NOP
адррес: команда
```

В простейшем случаи, программист должен обрабатывать это сам и вставлять NOP там где он нужен, однако можно использовать также оптимизирующий компилятор, который будет сам это делать. Но в лабораторных работах мы будем вставлять NOP'ы сами.

5.3.4 Порядок следования байтов

0-7	15-8	23-16	31-24
B0	B1	B2	B3

A0, A1, A2, A3 - адрес.

B0 - младший байт, B3 - старший байт.

Такой порядок следования называется Little-Endian (младший к старшему). Используется в процессорах Intel, HP и т.д.

Возможен и другой вариант – естественный порядок.

B3	B2	B1	B0

A0, A0+1, A0+2, A0+3

Big-Endian. IBM, Apple и т.п.

MIPS умеет переключать режимы.

5.3.5 Режимы работы

1. Kernel – режим ядра ОС (супервизор)
2. User – режим пользователя

Адресное пространство ОЗУ: 1ГБайт – только Kernel, 2ГБайта – Kernel/User.
При включении находимся в режиме Kernel
Переход в User – установка бита в регистр STATUS.
Возврат в Kernel – RESET или прерывание. Обработка прерываний выполняется всегда в режиме ядра.

5.3.6 Регистровая модель процессора

Регистры общего назначения:

31 0
r0
r1
r2
...
r31

31 0
HI
LO

31 0
PC

r0 = 0 – только чтение.
r31 – хранит адрес возврата при вызове подпрограмм.
HI, LO – для умножения/деления.

MUL (рез-т)	HI	LO

	остаток	частное
DIV (рез-т)	HI	LO

5.3.7 Система команд

Все команды 32 бита.
Проблема довольно серьезная: нужно в 32 бита вписать код операции, код адреса и непосредственный операнд:

Код операции	код адреса	операнд

Один только код адреса должен занимать 32 бита, но разработчики исхитрились это обойти: адрес команды всегда кратен 4, иначе – прерывание.

Команды загрузки/сохранения (обращение к ОЗУ)

- LB r_t, offset(r_b); байт со знаком в r_t (РОН).

31	7	0
знак	s	байт

- LBU r_t, offset(r_b);

31	7	0
0...0		байт

- LH r_t, offset(r_b); загрзука полуслова со знаком (16 бит).
- LHU r_t, offset(r_b); загрузка полуслова без знака.
- LW r_t, загрузка слова
- LUI r_t, Im; загрузка непосредственного операнда

31	7	0
IM	0..0	

Сохранение:

- SB r_t, offset(r_b); байт → ОЗУ
- SH ...; полуслово в ОЗУ
- SW ...; слово в ОЗУ

Арифметичекие операции

- ADD r_d, r_s, r_t; r_s+$r_t \rightarrow r_d$ Сложение со знаком.
- ADDI r_d, r_s, Im; $r_s + Im \rightarrow r_d$ Сложение со знаком. Im - 16bit.
- ADDU без знака
- ADDIU без знака
- SUB r_d, r_s, r_t; вычетание со знаком
- SUBU r_d, r_s, r_t; вычетание без знака
  Прерывание при переполнении! Флаги не устанавливаются.
- MULT r_s, r_t
- MULTU r_s, r_t
- DIV
- DIVU
- MFHI r_d – HI$\rightarrow r_d$
- MFLO r_d – LO$\rightarrow r_d$

Логические операции

- AND r_d, r_s, r_t;
- ANDI r_d, r_s, Im (16 разрядов)
- OR
- ORI
- XOR
- XORI
- NOR r_d, r_s, r_t ИЛИ-НЕ

Операции сдвига

- SLL r_d, r_s, n_s; $r_s \ll r_d$ (на n_s). ns = 0..31;
- SLLV r_d, r_s, r_t; $r_s \ll r_d$ (на r_t, младшие разряды)
- SRL
- SRLV
- SRA – аифметический вправо
- SRAV – арифметический вправо

Безусловный переход

- J addr; addr $\rightarrow$ PC

  Новый адрес:

31 28	27 2	1 0
PC	addr 28	00

- JR r_t; $r_t \rightarrow$PC
- JAL addr; $PC \rightarrow r31$; addr$\rightarrow$PC
- JLR r_d,r_s; PC$\rightarrow r_s$; $r_d \rightarrow$ PC
- Возврат из подпрограммы JR, r_s

Условные переходы

- BEQ r_s, r_t, offset16; переход если $r_s = r_t$
  Адрес перехода задается относительно – PC + offset16.
- BNE; $r_s \neq r_t$
- BLEZ r_s, off $r_s \leq 0$
- BGTZ ; $r_s > 0$
- BLTZ ; $r_s < 0$
- BGEZ ; $r_s \geq 0$

Ветвление с возвратом

- BLTZAL PC→ $r_3$1 ветвление если rs < 0
- BGEZAL PC→31, если $r_s \geq 0$

5.3.8 Сопроцессор управления (CP0)

Функции:

1. Трансляция адреса для страничной адресации.
2. Организация обработки прерываний

Страничную адресацию мы изучать не будем.
В сопроцессоре есть 12 дополнительных регистров: 6 для трансляции адреса и 6 для обработки прерываний.
Обращение к CP0: команды MTC0 rc, r1; MFC0 rc, r1 (Move to CP0, Move From C0)
Основные регистры:

1. config – конфигурация (r3)
   Полезные биты:
   (a) halt = 1 – остановка процессора (запуск reset или int)
   (b) RF=1 – снижение частоты в 32 раза. Уменьшает энергопотребление
   (c) Поле FInt – номер входа Int на который подается запрос FPU.
2. status – состояние
   (a) Биты C3-C0 = 1 – разрешен ли сопроцессор. (один из двух)
   (b) Int Mask (8 бит) – маска прерываний.
   6 входов запросы (из них один FPU), 2 програмных прерывания.
   (c) KU_c (для предыдущего KU_p, предпредыдущий KU_0) = 1 – user, 0 – kernel (режимы)
   (d) IE_c (для предыдущего IE_p, предпредыдущий IE_0), общая маска прерываний
3. cause – причина прерывания
   (a) Вектора исключений.
   Обслуживание исключений: PC → EPC, 0→KUc, 0→IEc KUc, IEc → KUp, IEp KUp, IEp → KU0, IE0 вектор Vi → PC
4. EPC – сохранение PC при прерывании
5. IWatch – адрес команды. По этому адресу нельзя читать – иначе перырвание
6. DWatch – адрес данных. По этому адресу нельзя писать – иначе прерывание

Причины прерываний

1. Ошибки трансляции адреса.
2. Ошибка шины. (сигнал Bus Error)
3. Ошибка адреса. (попытка записи из User в Kernel)
4. Команды прерываний
   - sys – системный вызов. System Call
   - bp – точка остановки – break point
5. Неправильный код команды
6. Обработку к недоступному сопроцессору CP1
7. Внешния прерывания Int – 6 входов запросов. 2 бита программного прерывания

Итого 16 входов прерываний, пронумерованных от 0 до 15. Чем больше номер, тем больше приоритет.

Вектор прерываний

- Reset – адрес 1FC00000. Ошибка страницы: 00..00, остаток 00..080.
  Этот вектор должен вызывать анализ регистров cause и status.
- Ошибка трансляции адреса – 00000000.
- Все остальные – 00000080.

5.4 Структура MIPS-системы

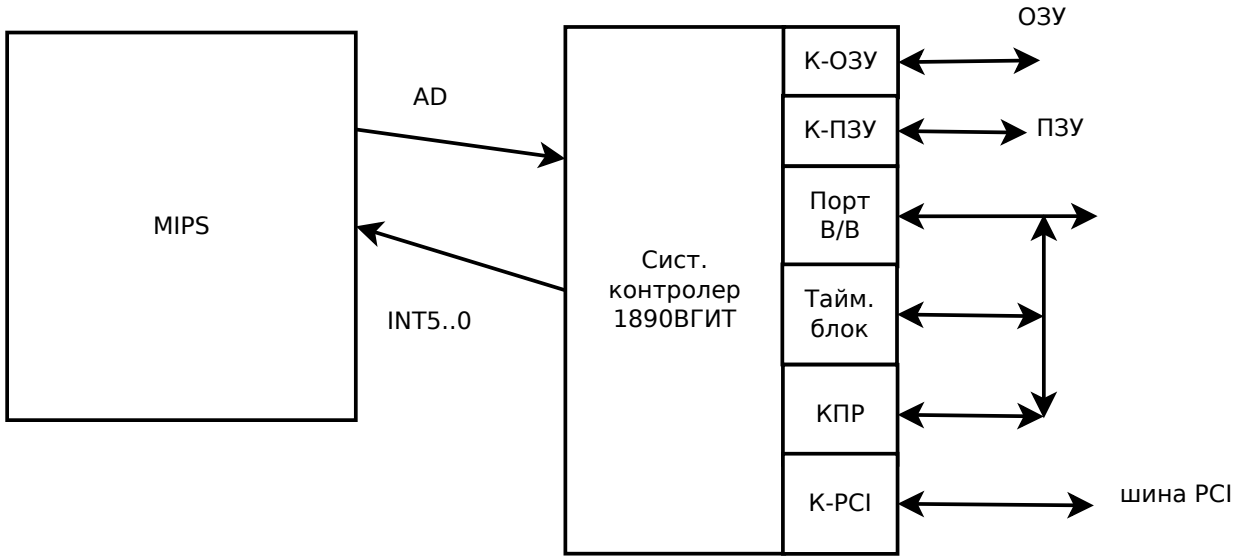


Рис. 5.2: MIPS

5.4.1 Состав СК

- Контролер ОЗУ
- Контролер ПЗУ (flash)
- Контролер PCI
- Параллельный порт (16 разрядов)
- Таймерный блок (3x32bit)
- КПР – контролер прерываний
- 3 последовательных COM-порта (RS-232)

Паралельный порт

Регистр входа: PioInReg. Выхода: PioOutReg
Одна из команд: LH ri, PioOutReg addr (SH)
Регистр направления: PioDirReg. 0 – ввод, 1 – вывод
Регистр функций PioFuncReg. 0 – ввод или вывод. 1 – альтернативные функции (запрос прерывания, выход т...)

Таймеры A,B,C

32 разряда. Реализуют прерывания. Внутренняя частота 4МГц или внешняя.
Load RegLo; Load RegHi – загрузка значения.
Read RegLo; Read RegHi – чтение текущего состояния.
Регистр состояния – TimerStateReg Когда счетчик = 1 – прерывание.

Биты состояния

- TS0 – режим однократной цикличности. 0 – цикличный, 1 – однократный.
- TS1 – остановка таймера.
- TS2 – разрешение прерывания.
- TS3 – флаг окончание (содержит 1 когда таймер дошел до 1).
- TS4 – прерывание(=0) или Reset(=1).
- TS5 – выбор синхронизации – внутреняя 4МГц или внешняя.
- TS6 – автоматическая перезагрузка модуля счета

Таймер A в циклическом режиме. Внутрений меандр.

Контролер прерываний

- Прерывания таймеров A,B,C
- Прерывания параллельных портов
- Прерывание динамической памяти
- Прерывание шины PCI.
- Ошибки обращения к памяти
- Прерывания последовательных портов SCI
- Внешние запросы (4 штуки)

К МП идет 4 входа: Int0-3, регистры IntLevel0, IntLevel1
IntLevel0 – IS7-IS0
IntLevel1: IS15-IS8
IS = 0, 1, 2, 3 – номер Int.
Регистр запросов IntRequst
Регистр разрешения IntEndle
Выходов 4 – устанавливаются вектора запросов.

5.5 RISC-процессоры PowerPC

5.5.1 Введение

1990 г. – IBM, Motorola, Apple объеденились и решили создать новый процессор. Сложилсь в сумме на 1 млрд. долларов, построили новый центр в Техасе, отправили 500 человек и через чуть больше года получили архитектуру PowerPC.

В наше время, PowerPC занимается только IBM и Freescale (бывшее подразеделение Motorola). Используются как высокопроизводительные процессоры.

В списках Top 100 - примерно 70% использует x86, а 30% - PowerPC.

Пример технических характеристик современных PowerPC. Частоты - 1.6ГГц, 2-х ядерный процессор имеет площадь 3.6мм^2, а потребляет 1.6Вт. Техпроцесс – 45нм.

5.5.2 Основы архитектуры PowerPC

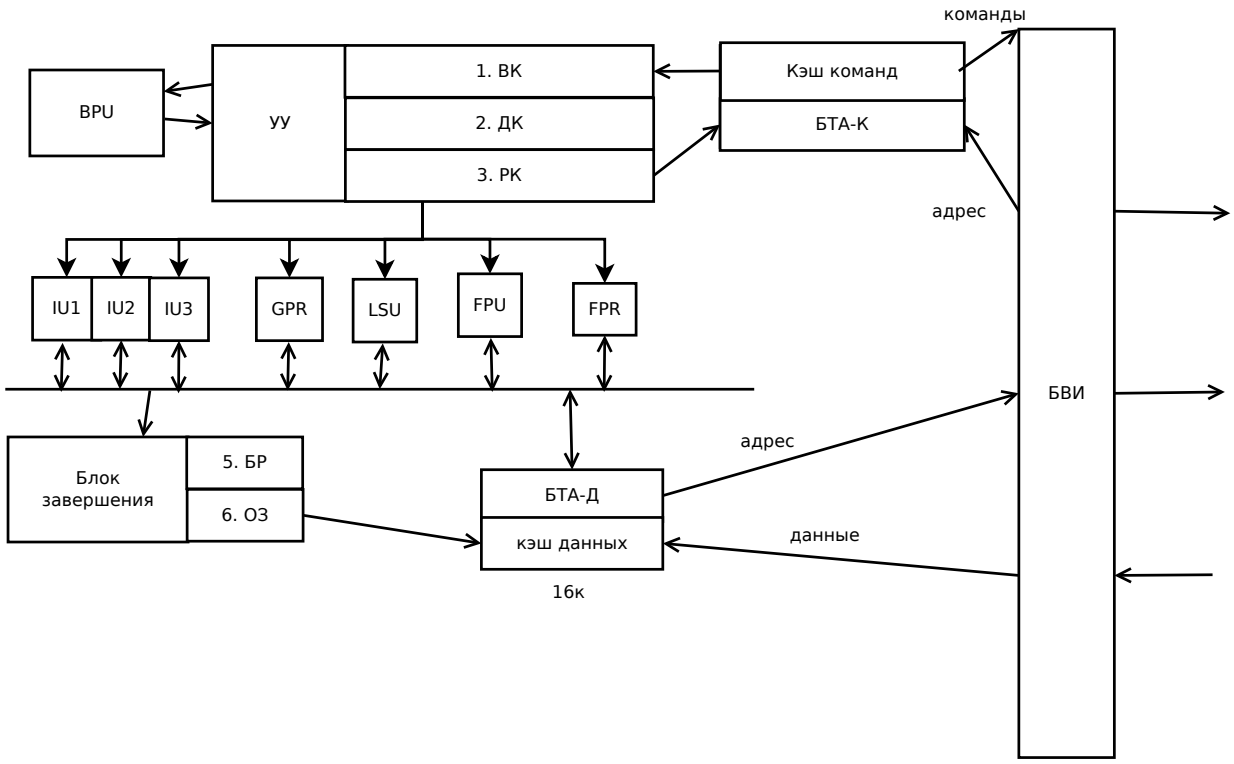


Рис. 5.3: PowerPC

- БВИ – Блок Внешних Интерфейсов.
- БТА – Блок трансляции адреса. Сегментная. Страничная.
- GRP – регистры общего назначения. 32 + 12x32p. Иначе называются РОП.
- FRP – регистры плавающей точки. 32 + 8x32p.
- Блок завершения – специальный блок, обеспечивающий восстановление есетственного порядка команд.
- BPU – блок обработки ветвлений. Предсказывает направление ветвления программы при поступлении соответствующих команд.
- IU1,2 – целочисленные операции, требующие 1 такт
- IU3 – реализует команды умножения (3-4 такта) и деления (20 тактов).
- FPU – операции с FP
- LSU – блок обращения к ОЗУ.
- УУ – Устройство управления.

Конвейер

1. ВК – выборка команд
2. ДК – дешифровка команд
3. РК – распределение команд (очередь)
4. ВП – выполнение команд
5. БР – буфер результатов
6. ОЗ – обратная запись (восстановление порядка результатов)

Особенности архитектуры

- Устройства IU, FPU, LSU, BPU работают параллельно.
- Архитектура – суперскалярная
- Из регистров общего назначения – 32 32-х разрядный и 12 для замещения, на случай параллельных операций. Аналогично с FP, только регистров замещения там 8.
- Два режима работы: User и Supervisor.

Регистровая модель USER

В регистровую модель пользователя, которая является общей для всего семейства PowerPC входят:

- 32 GPR (32 bit).
- 32 FPR для хранения операндов с плавающей точкой (64 бит).
- FPSCR – 32-х разрядный регистр управления FPU. **NOTE: источник: http://www.ibm.com/developerworks/linux/library/l-powarch**
- CR – регистр условий (признаков). 32 бит
- XER (FIxed-Point Exception Register), LR (Link Register), CTR (CounT Register), PC – 32-х разрядные.
- CR – регистр условий. Состоит из 8-и 4-битовых полей CR0-CR7. Мы можем таким образом хранить результат 4-х предыдущих сравнений.
- XER – регистр прерываний. Содержит признаки переноса CA (бит 1), переполнения OV (бит 2), устанавливается в результате выполнения арифметических целочисленных операций. Признак общего переполнения – SO (бит 0).

  В основном используются 3 младших бита. SO отличается от OV тем, что не сбрасывается при следующих операциях.
- LR – регистр связи. Служит для хранения адреса команды, к которой осуществляется условный или безусловный переход. При вызове подпрограммы сохраняет **адрес команды возврата**.
- CTR – регистр-счетчик. Содержит число циклов. Используется для организации программных циклов, при этом содержимое уменьшается на единицу после завершения очередного цикла.
- FPSCR – регистр управления состоянием FPU.

FPU осуществляет обработку чисел OT или ДТ, согласно стандарту IEEE754 (см FPU Pentium).

Режим supervisor

Регистровая модель режима **supervisor** включает в себя регистровую модель пользователя, а также содержит несколько групп служебных регистров.

Регистр управления MSR – определяет режим функционирования процессора. Можно задавать режим User/Supervis включение кэша, режим FPU, режим отладки и т.п.

Регистры, определяющие границы адреса, обработку прерываний, различные служебные регистры (таймеры и т.п.)

5.5.3 Способы адресации

- Регистровая
- Косвенно-регистровая со смещением
- Косвенно-регистровая с индексированием
- Относительная
- Абсолютная
- Непосредственная

Все команды обработки (арифметические, логические операции, сравнение, сдвиги) – регистровая или непосредственная. Это повышает производительность, так как при обработке данных не требуется выполнять циклов обращения к шине для выборки самих операндов.

Обращения к памяти – консвенно-регистровая со смещением (со знаком)(EA=rA+d32, при rA=0 EA = d32), индексная (EA=rA + rB, rA = 0 → EA=rB). rA, rB – номера регистров общего назначения, d32 – 32-х разрядное смещение.

5.5.4 Система команд

Формат команд (4 или 8 байт)

32 бита – код операнда, Im или d32. Хотя это и RISC, но команд более 200.
Следует помнить:

- rA, rB – номера регистров общего назначения. Регистры операндов
- rD – регистр, где размещается результат.
- Slm – непосредственный операнд со знаком
- Ulm – непосредственный операнд без знака
- d32 – 32-х разрядное смещение.

Команды арифметических операций имеют ряд модификаций, отличающихся формированием признаков по результатам операций. Если после мнемокода операции идет символ «˙», то по результатам операции устанавливаются соответствующие значения битов SO, EQ, GT, LT в поле CR0 регистра CR. Если имеется суфикс «o », то по результату выполнения устанавливаются биты OV, SO в регистре XER, при этом бит SO дублируется в поле CR0.

Команды загрузки-сохранения

Команды загрузки и сохранения производят загрузку в регистр GPR с номером rD байта, слова (32 разряда) или полуслова (16 разрядов) из памяти (команды начинаются с буквы l - load) или записывают в память байт, слово или полуслово из регистра rs (команды, начинающиеся с буквы st – store).

- lbz rD, d(rA); (EA) → rD; Загрузка байта.
- lbzx rD, rA, rB; (EA)→rD (последующее использование EA)
- аналогично lhz, lhzx, lha, lhax, lwz, lwzx.

lha – алгебраическое (the remaining bits are filled with a copy of the most-significant bit of hte loaded half-word). lhz – half-word and zero (remaining bits are filled with zero).
lbzx – индексная адрессация (EA = rA + rB)

Сохранение

- stbz rS, d(rA) rs→EA
- sthz, sthzx; полуслово
- stwz, stwzx; слово

Групповая загрузка

- lmw rD, d(rA); (EA)→rD...GRP31. Групповая пересылка.
- stmw rS, d(rA); rS...GRP31→ (EA).

Арифметические операции

- add rD, rA, rB; rA+rB→rD
- addi rD, rA, Im; rA+Im→rD
- add. – установка признаков в CF0
- addo – контроль переполнений

Команды управления

- b addr; addr→PC. Безусловный переход
- ba rel; rel+PC→PC
- bl addr; PC→LR, addr→PC. Безусловный с возвратом.
- bla rel; PC→LR, (PC+rel)→PC.
- bcz BO,CRi, addr; Условный переход. Если BO=CRi, addr→PC. BO – 4 бита условия сравниваются с CRi. 5-ый бит = 1 – предсказание ветвления.
- bcl BO,cfi,rel; с возвратом.
- tw TO,rd,rs; сравнение rd-rs, установка CR0, сравнение T0 с CR. Программное прерывание. TW – trap word
- rti; возврат.

  SRRO, SRR1 → PC, MSR. При T0 = CF0 PC→SRR0, MSR→SRR1.

Примечание: SRR0, SRR1 – Status save/Restore Register, служат для сохранения контекста и адреса возврата при прерываниях. Автоматически восстанавливают информацию при возврате из прерывания.

5.6 ARM

Advanced RISC Machines (1995 г.)

5.6.1 Разработка

- Модель на VHDL/Verilog

  Сам синтезируй и вставляй в модуль.
- Электрическая схема (Netlist)
- Топологический фрагмент.

5.6.2 Разновидности

Существует несколько резновидностей:

- ARM7
- ARM9.

  Кэш до 16КБ. Есть MMU (Memory Management Unit – блок трансляции адреса)
- ARM11 – кэш больше, до 1МБ. Есть FPU (VFP), DSP.

	0.18мкм			0.15мкм		
Название	S	P	F_t	S	P	F_t
ARM7	0.6 мм^2	$0.3\frac{\text{мВт}}{\text{МГц}}$	до 110	0.3 мм^2	$0.1\frac{\text{мВт}}{\text{МГц}}$	до 130
ARM9	4 мм^2	$0.9\frac{\text{мВт}}{\text{МГц}}$	до 250	2 мм^2	$0.2\frac{\text{мВт}}{\text{МГц}}$	до 320

Табличка из других лекций

Ядро	Техпроцесс	MIPS/MHz	мВт/МГц	S, мм^2	F, МГц
ARM7	0.18, 0.13	0.9-1	0.4, 0.1	0.6, 0.3	80-110, 100-133
ARM9	0.18, 0.13	1.1	0.8-1, 0.2-0.3	4-5, 2-2.5	150-200, 180-270
ARM11	0.13	1.2	0.6	6-10	200-330

5.6.3 Особенность

- 2 режима работы:

  ARM – 32 разярдные команды и данные

  THUMB – 16 разрядные команды, 32 разрядные данные

- 3-х ступенчатый конвейер: 1 команда за 1 такт. Условие выполняется для всех команд

- Режимы работы:

  1. User – пользователь.
     Регистровая модель: доступны R0-R12, R13(SP), R14(LR), R15(PC), CPSR
  2. Supervisor – ОС
     Регистровая модель: доступны R13, R14, PC, CPSR, SPSR.
  3. System – Supervisor + данные User
  4. IRQ – обработка прерываний
     Регистровая модель: R13, R14, PC, CPSR, SPSR.
  5. FIQ – Fast IRQ
     Регистровая модель: доступны R8f-R14f, PC, CPSR, SPSR (копия CPSR из User)
  6. Undefined – неправильный код команды
     Регистровая модель: R13, R14, PC, CPSR, SPSR.
  7. Abort – ошибка обрещения к памяти
     Регистровая модель: R13, R14, PC, CPSR, SPSR.

Основная задача – упросить процедуру переключения.

Процедура включения

Включение, Reset→supervisor→user
User не может переключить в Supervisor.
Всего 16 регистров, в том числе счетчик. Описания:

- R15 – PC
- R14 – LR, регистр связи. Сохраняет адрес возврата.
- R13 – SP – указатель стека. НА самом деле обычный регистр, который лишь рекомендуется использовать как указатель стека, если стек вообще используется.

R14 и R13 дублируются для каждого режима (отдельно для каждого, именно поэтому и рекомендуется использовать R13 для стека).
CPSR – регистр состояния.
SPSR – копия CPSR.

Режим FIQ

Регистры R8-R12 дублируются. Происходит сохранение контекста. Возможно быстро переключиться.

Регистр CPSR

- N,Z,C,V – признаки как и везде (N – знак, Z – 0, V – переполнение, C – перенос)
- I – IRQ
- F – FIQ
- T – Thumb(1)/ARM(0).
- M – режим. 10000 – User, 10011 – Supervisor. Остальные режимы устанавливаются автоматически при прерываниях или ошибках.

При переключении режима CPSR копируется в SPSR.

5.6.4 Адрессация

- Обработка данных
  - Регистровая
  - регистровая с масштабированием (со сдвигом)
    Shift, #n; n – число разрядов сдвига (0-15). Сдвиг идет на 2n, т.е. можно сдвинуть на 30 раздядов.
    Shift: LSL (логический влево), LSR (лог. вправо), ASR (арифметический вправо), ROR (циклический вправо)
    То есть мы можем делать пересылку операндов со сдвигом.
  - Непосредственная
- Обращение к памяти. Команды загрузки-сохранения
  - Косвенно-регистровая (Rn)
  - Косвенно-регистровая со смещением. Rn, +offset. Offset – 12bit. Адрес=Rn±offset
  - Индексная (Rn, ± Rm), адрес = (Rn)±(Rm)
  - Индексная со сдвигом (масштабированием) (Rn, ±Rm shift #n)
  - Пост-индексная
    (Rn), ± Rm; Адрес = (Rn), после выборки Rn±Rm→Rn
  - Пост-индексная со сдвигом
    (Rn), ±Rm shift #n
    После выборки Rn=$Rn \pm Rm$ shift #n

5.6.5 Система команд

Как и в большинстве RISC комманд мало.

Пересылка

- LDR(cc) Ra,addr; загрузка слова. cc – условие
- LDRB(S) ...; загрузка байта с расширеннием нулями (если S – расширение знаком)
- LDRH(S) ...; загрузка полуслова с расширением нулями (если S – знаком)
- STR(cc) Rd,addr ; сохранение слова
- STRB
- STRH
- LDM(cc) Type,Rn, Rm – групповая загрузка. Начальный адрес Rm списка регистров. Type указывает на то, как будет меняется номер:

Суфикс	Описание
IA	Increment After
DA	Decrement After
IB	Increment Before
DB	Decrement Before

- STM(cc) Type, Rn, Rm; Групповое сохранение.
- MOV(cc) Rd Opr; Opz→Rd, Opr=Rs, (Rs) shift, Im. Любое из 3-х можно задать cc.
- MRS(cc) Rd, CPSR/SPSR; CPSR/SPSR→Rd
- MSR(cc) CPSR/SPSR, Rm; Rm→CPSR/SPSR

Условия (cc):

Суфикс	Условие	Комментарий
EQ	Z = 1	Равно (Zero set)
NE	Z = 0	Не равно (Zero cleared)
CS	C = 1	Carry Set
CC	C = 0	Carry Clear
MI	N = 1	Minus
PL	N = 0	plus
VS	V = 1	overflow set
VC	V = 0	overflow clear
HI	C=1, Z=0	Higher then (без знака)
LS	C=0, Z=1	Lower then (=CC)
GE	N=V	$\geq$, со знаком
LT	N$\neq$V	$<$, со знаком
GT	Z=0, N=V	$>$
LE	Z=1, $N \neq V$	$\leq$
AL		Всегда. Можно опустить.

Арифметические операции

Особенность является отсутствие операции деления.

- ADD (cc) (s),Rd,Rn,Opr; Opr+Rn→ Rd; s=1 установка признаков/
- ADC (cc) (s),Rd,Rn,Opr; то же, с учетом переноса
- SUB (cc) (s),Rd,Rn,Opr; вычетание
- SBC ...; вычетание с учетом переноса.
- MUL (cc) (s),Rd,Rn,Rs; $Rn \times Rs \to Rd$. Младшие 32 бита. s – установка признаков
- (S,U)MUL (cc), (s), Rdh, Rdl, Rn,Rs; $Rn\times \to Rdh : RdlRs$. 32x32 в 64 бита. S – со знаком, U – без знака.
- MLA(cc) (s) Rd, Rm, Rs, Rn; Умножение с накоплением. $(Rm \times Rs) + Rn \to Rd$. Младшие 32 разряда.
- (S,U)MLA (cc) (s) Rdh,Rdl,Rm,Rs; $(Rm \times Rs) + (Rdh : Rdl) \to Rdh : Rdl$. 32x32 в 64.

Логические операции

- AND(cc)(s) Rd,Rn,Opr; Rn AND Opr → Rd
- ORR ...
- EOR ...
- BIC ...; Rn AND $\overline{Opr}$ →Rd

Команды управления

- B(cc) rel. rel – 24 бита. PC + offset → PC. Смещение со знаком.
- BAL rel; безусловное ветвление.
- BL(cc) rel; подпрограмма. L – link. $PC \to LR$, $PC + rel \to PC$.
- BX(cc) Rn; переход со сменой режима:

  Rn:

31 2	1	0
offset		T

  T=1 – Thumb, T=0 – ARM.

Программные прерывания

Переход в режим Supervisor

- SWI(cc) cmnt; cmt – комментирий, можно не использовать.

  PC→LR

  $V_c = \$0008 \rightarrow PC$

  0008 – фиксированный вектор для программных прерываний.

  Для выхода из прерываний нет специальной команды. Для программных используется MOV R14,R15; копирование LR в PC.

  Для прерываний используется MOVS R14, R15; SPSR→CPSR; $LR \rightarrow$PC

Глава 6

Операционные системы реального времени (ОСРВ)

6.1 Введение. Базовая теория.

Операционная система – совокупность программ, управляющих оборудованием.
Операционная система – совокупность программ, управляющих другими программами.
Довольно большой класс устройств не использует ОС, в частности микроконтроллеры. ОС нужны если:

1. Если вычислительная система используется для различных задач, при этом программы, исполняющие данные задачи, нуждаются в сохранении данных и обмене данными между собой.

   Есть ряд задач для которых используется ВС. Для решения этих задач написан ряд программ, делающих определенные действия. Программам нужно обмениваться данными, а, значит, нужен механизм сохранения – какая то файловая система, позволяющая сохранить данные в любую энергонезависимую память (HDD, Flash), а также может понадобится осуществлять обмен данных без сохранения на диск.

2. различные программы нуждаются в выполнении одних и тех же действий.

   Чтобы не дублировать какой-то код (считывание символов, вывод на экран и так далее), ОС может предоставлять набор библиотек. Это необходимо когда есть несколько программ, для одной программы это не нужно.

3. Между программами и пользователями необходимо распределять полномочия.

   Идея тут такая: защита системы от тотального сбоя, если сбой произошел только в одной программе. Надежность системы в целом повышается.

   Также данные одних пользователей защищаются от других пользователей.

4. Необходима имитация параллельного исполнения нескольких программ.

   Один процессор, нужно использовать более чем одну программу. Нужна служба «Планировщик задач», которая распределяет процессорное время между программами (кванты системного времени). Поскольку квант времени маленький, создается иллюзия параллельного исполнения задач. Про планировщик мы поговорим отдельно.

5. Пользователю необходимо иметь возможность управлять исполняющимися задачами.

   Любая ОС реализует набор системных утилит (базовое ПО) и оболочку (командный интерпретатор, гуй).

Основные функции ОС:

1. Загрузка приложений в оперативную память и их выполнение.

   Приложение в рабочей системе при отключенном питании хранятся в энергонезависимой памяти, но при запуске загружается ОС, загружает какие-то приложения и передает им управление.

2. Стандартизированный доступ к периферийным устройствам.

   Это достигается как правило с использованием драйверов (модулей ядра), которые зная особенности того или иного оборудования, реализуют стандартизированный интерфейс. Например у нас есть USB мышь и RS-232 мышь, но ПО работает с ними одинаково.

3. Управление оперативной памятью.

   Распределение ОЗУ между процессами в системе. Процессов много, память одна. Должен быть какая-то общая точка входа – тот кто контролирует текущее использование и обработчик запросов на выделение и освобождение памяти.

4. Управление доступа к данным на энергонезависимых носителях (реализация файловой системы).

   Файловые системы бывают разные. Как правило для каждой ОС есть своя нативная (родная) ФС, которая разрабатывалась для этой ОС (Windows 95/98 - FAT32, Windows NT - NTFS, Linux - ext2/3). Есть также специализированные файловые системы для конкретных нужд, например jffs для flash. jffs позволяет минимизировать количество реальных циклов стираний. Во flash-памяти каждый раз когда мы хотим поменять данные мы должны считать сектор, модифицировать его, стереть и записать новый. jffs дополняет информацию некоторыми служебными данными, а затем дописывает в свободное место – увеличивает время жизни флешки ценой скорости.

5. Пользовательский интерфейс.

6. Сетевые операции и поддержка стека протоколов.

   7-и уровневая модель OSI. За реализацию отвечает модуль ОС

7. Параллельное выполнение задач.

8. Взаимодействие между процессами (IPC – interprocess communication).

   Организация ряда способов, с помощью которых данные от одного процесса могут попасть в другой процесс, а также синхронизация.

9. Защита самой ОС, а также данных и программ пользователя от злонамеренных действий других пользователей.

   Пользователи бывают плохие и хорошие. Плохие пользователи иногда хотят сделать бяку хорошим. Бывает, что программа из-за ошибки может начать работать неправильно, а нужно чтобы не страдали другие программы.

10. Разграничение прав доступа и многопользовательский режим.

    Требуется некоторая иерархия ОС.

    Обычно имеется такая иерархия: ядро -> системные библиотеки -> оболочка с утилитами.

    Ядро включает в себя планировщик (scheduler), драйверы (drivers), сетевой стек (network), таймеры (timers).

Применение универсальных компьютер для управления технологическими (производственными) процессами потребовал реализации реального масштаба времени, то есть синхронизации исполняемых программ с внешними физическими процессами, что привело к появлению нового типа ОС – ОСРВ.

По стандарту POSIX 1003.1 (POSIX – Portable Operating System Interface for UNIX) реальное время в ОС это способность ОС обеспечить требуемый уровень сервиса (обслуживания) в определенный промежуток времени.

ОСРВ – такие системы, в которых правильность работы системы зависит не только от правильности полученного результата, но и от времени, за которое этот результат бы получен.

ОСРВ – ОС, реагирующая в предсказуемое время на непредсказуемое появление внешних событий.

6.2 Архитектура ОСРВ

Существует много архитектур, но их можно разделить на несколько типов:

1. Монолитная
2. Уровневая
3. С микроядром.

6.2.1 Монолитная архитектура ОСРВ

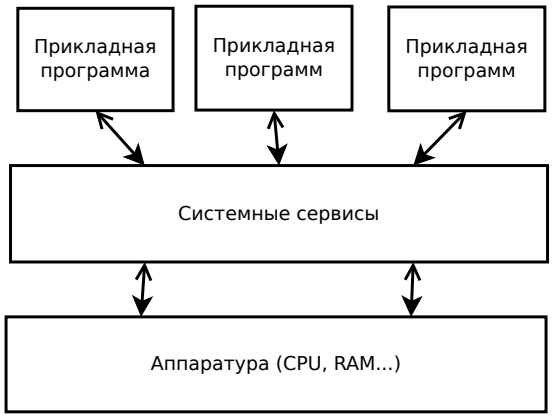


Рис. 6.1: Схема монолитной архитектуры ОСРВ

Предоставляют API – Application Programming Interface для взаимодействия с сервисными ядра. Плюсы:

- Высокая скорость работы
- Упрощенная разработка модулей ядра

Минусы:

- Одноадрессное пространство => сбой в одном модуле – сбой системы.

Наиболее известный представитель – LynxOS.

6.2.2 Уровневая архитектура ОСРВ

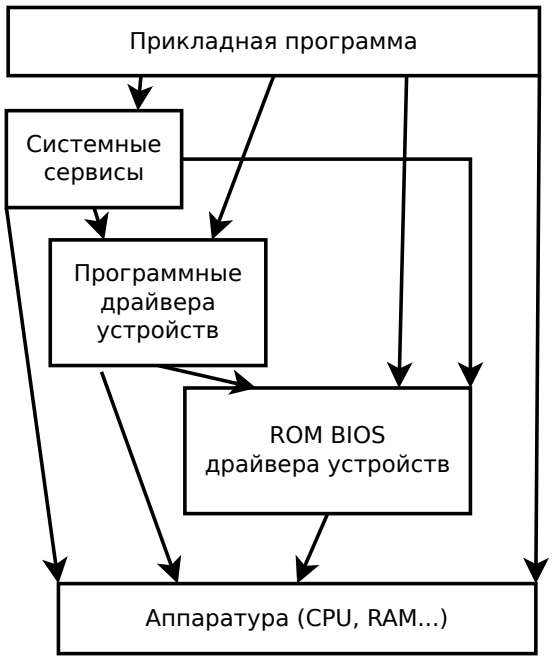


Рис. 6.2: Схема уровневой архитектуры ОСРВ

Плюсы:

- Скорость работы выше монолитной (прямой доступ к аппаратуре)

- Большая надежность

Минусы:

- Однозадачность (нет синхронизации)

Представитель: DOS

6.2.3 ОСРВ с микроядром (клиент-серверная)

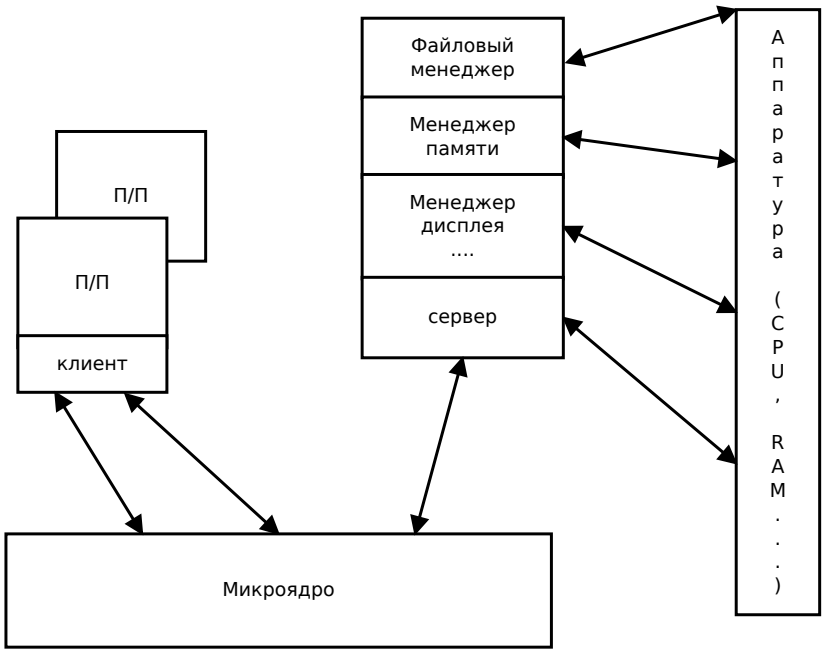


Рис. 6.3: Схема микроядерной архитектуры ОСРВ

Плюсы:

- Стабильность

Минусы:

- Накладные расходы

Представитель: QNX, VxWorks, ...
Самая популярная архитектура

6.3 Типы ОСРВ

- Жесткого реального времени (hard)

  Не допускают задержек реакции системы, так как это может привести к потери актуальности результатов, человеческим жертвам, чрезмерным финансовым потерям (что ещё печальнее) и прочим печальным последствиям.

  Примеры систем: системы управления бортового оборудования, системы аварийной защиты, регистраторы аварийных событий и так далее.

- Мягкого реального времени (soft)

  Характеризуются возможностью задержки времени реакции.

  Пример: роботизированный конвейер. Задержка может привести к необработанной детали, но это не приведет к фатальным последствиям (собирался автомобиль, а из-за задержки получились жигули).

Система жесткого реального времени никогда не опоздает с реакцией на события, а система просто не должна опаздывать на события.

6.4 Сравнение универсальных ОС и ОСРВ

Ключевое различие между ОСРВ и универсальной ОС – необходимость детерминированного поведения ОСРВ. Для любого сервиса ОСРВ время представляется алгебраической формулой, в которой не будет случайных временных величин. При этом если спросить разработчика универсальной ОС за какое время происходит передача сообщений, то такую информацию получить невозможно принципиально, а для ОСРВ можно. В ОСРВ **для каждого** действия можно написать формулу и, как правило, она выглядит так: T=const. Так передача сообщений всегда передается за фиксированное время вне зависимости от размера сообщения и количества процессов. Обычно аналогично с переключением контекста.

	ОСРВ	Универсальная ОС
Основная задача	Успеть среагировать на события, происходящие на оборудовании	Оптимальное распределение ресурсов между пользователями
Ориентация	Обработка внешних событий	Обработка действий пользователя
Позиционирование	Инструмент для создания конкретного программно-аппаратного комплекса	Набор приложений, готовых к использованию
Предназначение	Квалифицированный разработчик	Пользователь средней квалификации

Таблица 2001 года с форума RealTime and Embedded Systems Forum, представлена компанией ≪OSE Ssystem≪. Данная вещь является статистикой и поэтому слепо доверять ей нельзя.

Атрибут	Обычное ПО	Информ. системы	Оборонка	Телеком.	Медицина	Авионика
Размер ПО (LOC – Lines of Code)	100к-10М	1М	50к-5М	500к	10к-50к	10к-1М
Время разработки	1 год	4 года	15 лет	3-5 лет	3 года	7-10 лет
Время вывода на рынок	8 мес	1.5 года	6 лет	1 год	1.5 года	2-3 года
Стоимость небольшой ошибки, $	0	10к	100к-500к	0-100к	0-100к	100к-500к
Стоимость серьезной ошибки, $	минимальная	10к-500к	до 20М	500к (репутация)	100к-5М	1к-500М

Стоимость – сколько было потрачено на устранение последствий.

ОСРВ используется в Обороне, телекоммуникациях, медицине и авионике. Наибольшая ответственность в обороне, поэтому огромное количество времени уходит на тестирование всего, вплоть до мелочей.

6.5 Наиболее распространенные ОСРВ

Можно разделить на два класса:

- Свободное ПО (free).
- Собственническое ПО (Proprietary).

Свободное ПО обладает четырьмя степенями свободы:

1. Запускать
2. Улучшать
3. Изучать
4. Распространять

Свободное ПО всегда в открытых исходных кодах.
Распространяется обычно в виде исполняемых файлов.
Наиболее распространенные – свободные ОСРВ.

6.5.1 Свободные ОСРВ

1. RTLinux (realtime Linux).

   На базе обычного Linux, но реализовано микроядро, а основное ядро идет как процесс.

2. RTEMS – RealTime Executive for Multiprocessor Systems (ныне)

   Раньше было for Military Systems, а ещё раньше Missile Systems.

   Разрабатывалась по заказу мин. обороны США. Исторически сложилось, что популярна в аэрокосмической сфере.

   Пример использования: NASA, Mars Reconnaissance (разведка, исследование) Orbiter, радиомодуль Electra.

3. ОС2000 (Багет) – министерство обороны РФ (НИИСИ РАН).

   Для процессоров MIPS, Intel. Используется или нет – неизвестно.

6.5.2 Проприетарные ОСРВ

1. QNX/Neutrino

   Управление светофорами в Канаде.

2. LynxOS – авиация, управление технологическими объектами.

3. SymbianOS – мобильные телефоны.

4. VxWorks/Tornado – разработка WindRiver (США). Одна из самых дорогих. Одна из самых старых. Портирована на множество архитектур.

   Используется в NASA Phoenix Mars Lander; автомобили BMW

5. Windows CE.

   Не имеет ничего общего с Windows, кроме стабильности.

   Часто можно видеть очередь из-за зависшего кассового аппарата, если приглядеться, то часто можно заметить логотип Windows CE.

   Портирована на архитектуры x86, MIPS, ARM, Hitachi SuperH и т.п.

Преимущество проприетарных ОСРВ – сертификация.

6.6 Ядро ОСРВ

Ядро – часть ОС, предоставляющая большинство основных служб (функций) для прикладного ПО, работающего на процессоре.

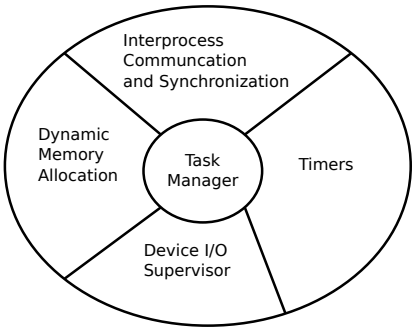


Рис. 6.4: Схема ядра ОСРВ

1. Task Management – управление задачами
2. IPCaS – взаимодействие и синхронизация задач. (синхронизация – обеспечить чтобы несколько процессов не работал с одним ресурсом)
3. Динамическое распределение памяти
4. Управление задачами ввода-вывода – унифицированный интерфейс
5. Управление таймерами

6.7 Распределение задач по времени (планирование задач)

6.7.1 Priority-based preemtive scheduling

Планирование с приоритетным прерыванием работы.

Допустим у нас есть несколько задач. Для каждой выделенной задачи назначается приоритет P, чем более важная задача, тем выше приоритет. Приоритет это некое число. Иногда чем больше число, тем выше приоритет, но бывает и наоборот.

Рассмотрим пример:

$$P(T_1) < P(T_2) < P(T_3); t(T_1) = 30\text{мс}, t(T_2) = 20\text{мс}, t(T_3) = 10\text{мс} \text{ -= время работы задач}$$
$$t_0(T_1) = 0\text{мс}; t_0(T_2) = 20\text{мс}; t_0(T_3) = 30\text{мс} \text{ – время запуска задач}$$

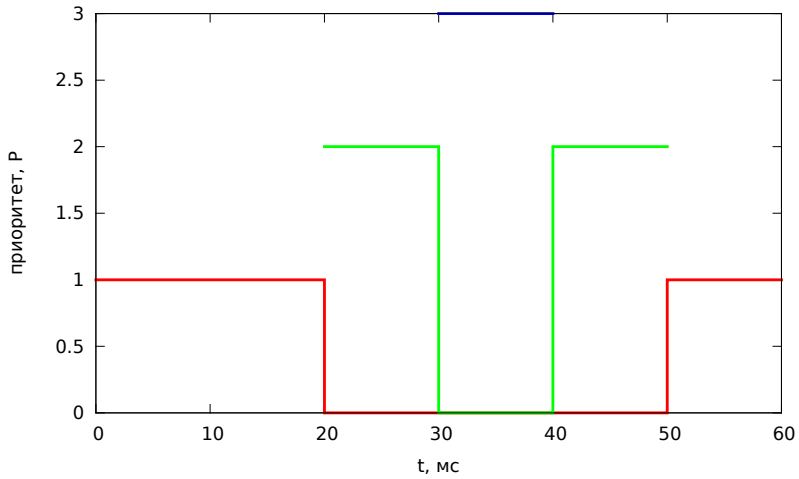


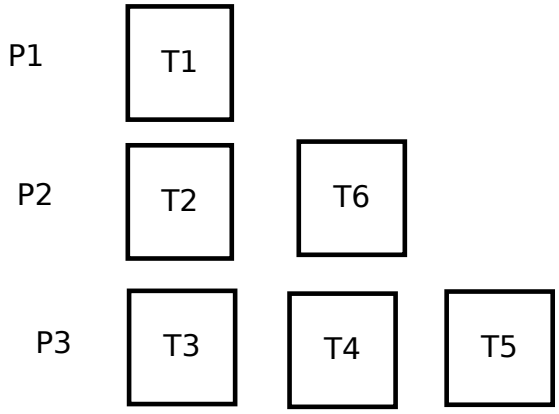
Рис. 6.5: Идеализированный график работы

На самом деле все несколько хуже. Существует время задержки Δt_p потому что планировщик делает примерно так:

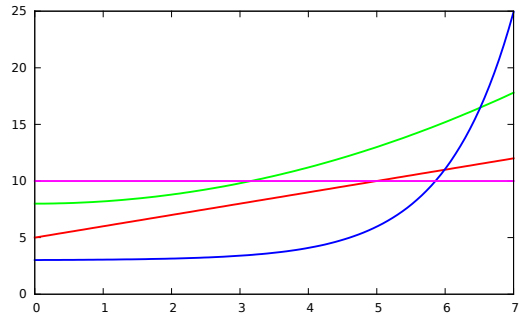
1. Должна ли текущая задача продолжать вычисление? Если да $\Rightarrow$ не переключаюсь. Если нет:
2. Определить какая задача должна выполняться следующей?
3. save current context (сохранение условий рабоыт предыдущей задачи)
4. restore new context (подготовить условия работы для новой задачи)
5. Передать управление новой задаче

Все эти пять шагов называются переключением задач.

Неопределенность может возникнуть на втором шаге, её необходимо исключить. В ОСРВ используются пошагово обновляемые списки (таблицы/очереди):

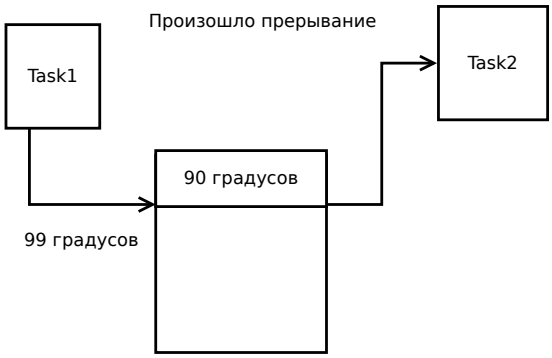


То есть нам нужно посмотреть только один список и взять первую (организуются разные списки для разных приоритетов)



Главное преимущество ОСРВ – постоянство. Универсальные ОС могут выполнять переключение задач быстрее, но только ОСРВ гарантирует предсказуемость времени переключения.

6.8 Синхронизация и обмен информацией между задачами



В результате рассинхронизации передаваемые данные могут искажаться. Для того, чтобы этого избежать, в ОС предусмотрены механизмы синхронизации.

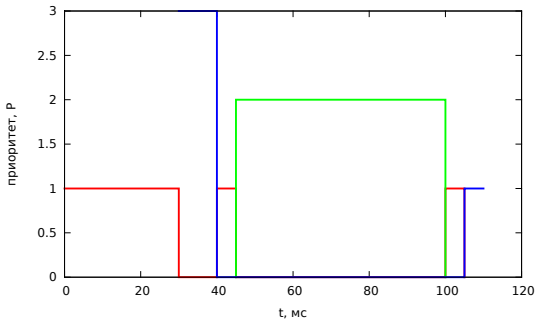
Синхронизация бывает двух типов:

- Отрицательная (семафоры, мьютексы, условные переменные и т.д.)
- Положительная (сигналы)

6.8.1 Отрицательная синхронизация

Отрицательные синхронизации используются для блокировки ресурсов.

При отрицательной синхронизации, может происходить инверсия приоритетов:



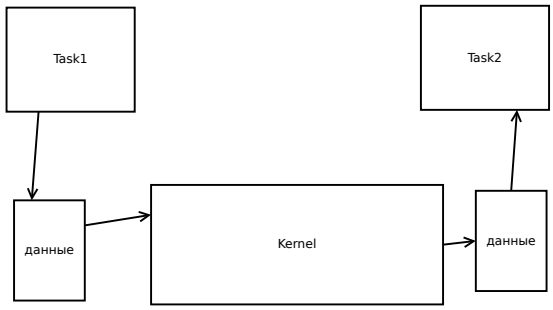
1. t_0 – T_1 захватывает ресурсов

2. t_1 – активизируется T_3
3. t_2 – T_3 попыталась захватить ресурс, которым обладает T_1. Захватить его она не может (в данном случаи приоритет не играет никакой роли). Значит нужно ждать пока ресурс освободится.
4. t_3 – T_2 – запустилась, ей ресурс не нужен и она продолжает работать до t_4, т.к. $P(T_2)$
5. t_4 – управление обратно в T_1
6. t_5 – T_1 освобождает ресурс и активируется задача T_3

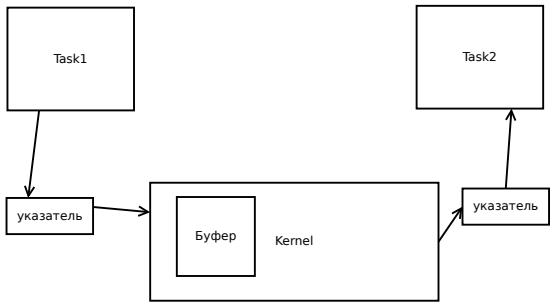
В таком случаи получается, что T_3 оказалась заблокировано задачей T_2 с более низким приоритетом (поэтому и называем инверсией) на $delta = t_4 - t_3$. Решается это следующим образом: временно приоритет задачи, заблокировавшей ресурс, повышается до приоритета блокируемой.

6.9 Передача данных

Ещё одна возможная область с недетерминированностью.
В универсальных ОС:



Есть ядро, которое знает все.
Но задачи передают данные друг-другу промежуточным копированием в область ядра.
В универсальных ОС время передачи сообщения от одной задачи к другой задаче зависит от размера сообщения, поэтому возникает недетерминированность.
В ОСРВ (сейчас так делают и в универсальных ОС):



$t = const$, т.к. указатель фиксирован. Буфер выделяется в пространстве ядра.

6.10 Динамическое распределение памяти

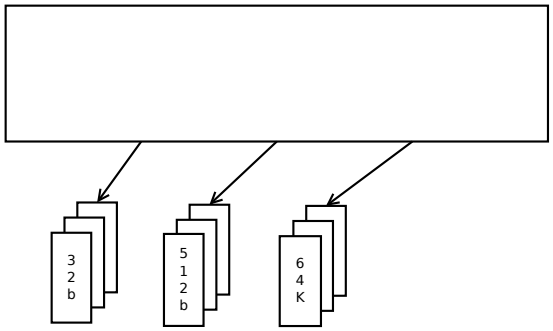
В универсальных ОС, как правило, используется heap(куча). malloc()/free() работают с heep.
Если задаче нужен какой-то кусок памяти она забирает её с помощью malloc, а когда становится ненужным – освобождает с помощью free. Здесь существует эффект дробления (эффект фрагментации).
Рассмотрим пример:

1. Пустая куча
2. p1 = malloc(128); p2 = malloc(128);
3. free(p1);

4. p3 = malloc (100);

Остается неиспользуемый кусок памяти. Возникает ситуация, что у нас есть как-бы свободная память, но которой нельзя пользоваться. Для этого в универсальных ОС используется специальную службу – сборщик мусора (garbage collector). Сколько времени будет работать сборщик мусора предсказать невозможно ⇒ недетерминированность.

В ОСРВ используется нефрагментируемые схемы работы динамической памяти:



Память разделяется на области фиксированных размеров.

Делают обычно не поровну – блоков с небольшими размерами обычно больше.

Свободная RAM делится на набор буферов фиксированных размеров и разрешаются выделять только буфер одного из фиксированных размеров.

Таким образом достигается детерминированность времени.

6.11 ОС-2000

ОС-2000, разработка НИИСИ РАН.

6.11.1 Архитектура

Независимая от аппаратуры (Си)	системные библиотеки, ...
Зависимая от CPU (Си, ASM)	функции работы с контекстом, пролог, эпилог
Пакет поддержки модуля (зависит от аппаратуры) (Си, ASM)	драйверы... (BSP)

Данная ОС поддерживается на MIPS, x86.

6.11.2 Процессы и потоки

Процесс – process, task – экземпляр исполняющейся программы.

У каждого процесса свое адресное пространство своя таблица ресурсов. Каждый процесс имеет хотя бы один поток (thread). По сути поток – тот же экземпляр процесса, но делающий другие действия (процесс клонируется, затем изменяется для выполнения других задач в рамках данного процесса).

6.11.3 Временные характеристики

- Время ответа на прерывания – время между запросом на прерывание и моментом, когда начала исполняться первая инструкция обработчика. Это время включает в себя время задержки прерывания. Задержка прерывания – время на которое в ядре запрещаются прерывания.
- Время ответа потока управления – время между моментом возникновения прерывания и моментом когда начала исполняться первая инструкция потока, который должен среагировать на это прерывание. Время ответа потока управления всегда больше потому, что оно включает в себя время ответа на прерывание

Архитектура	Время переключения контекста	Время ответа на прерывание	Время ответа потока управления
MIPS	$<$ 1мкс	$<$ 1мкс	$<$ 5мкс

6.11.4 Средства разработки

Инструментарий стоит на ЭВМ (компилятор, линковщик и т.п.).

После сборки, прошивка загружается по RS-232(zmodem) или Ethernet (TFTP) загружается на плату. Плата называется целевой ЭВМ (target), а компьютер на котором происходит разработка – инструментальная ЭВМ (host).

6.11.5 Потоки управления

В ОС-2000 в системе всегда один процесс и несколько потоков. Это значит, что не применяется виртуальная память.

$intpthread\_create(pthread\_t * thread, pthread\_attr_t * attr, void(*start\_routine)(void*), void * arg)$;

- *trhead – указатель на дескриптов потока (выходной параметр)
- *attr – аттрибуты потока (политика планирования, приоритеты...)
- *func – функция исполняющаяся в рамках потока
- *arg – аргумент передаваемый функции func при её вызове.

```
 static void my_thread(void *param)
{
        do {
        ...
        } while (1);
}

int main (int argc, char *argv[])
{
        pthread_t pdsc;
        int rw;

        rw = pthread_create(&pdsc, NULL, my_thread, NULL);
        if (rw == 0) {
                /* Thread created successfully */
                ...
        }
        return 0;
}
```

6.11.6 Планирование потоков

Есть три способа планирования:

- SCHED_FIFO – First In, First Out – приоритетное планирование.
- SCHED_RR – RoundRobin – приоритетное с разделением по времени.
- SCHED_OTHER == SCHED_RR

FIFO от RoundRobin отличается тем, что в системе реализуются 10мс кванты. При политике SCHED_RR управление может быть передано другие задаче с тем же приоритетом, что и прерывания по истечению кванта.

Поискать что такое mutex и чем отличается семафор от mutex.

Добавка от 11.01.2010: На пальцах spinlock (спинлок) можно описать как верную девушку. Занята так занята.

mutex – рассчетливая девушка. Занята так занята, но в уме держит всех остальных претендентов.

Семафор – нууу.... n-местный мьютекс (н - задается для каждого семафора).

Глава 7

DSP

Цифровая обработка: усиление, модуляция, фильтрация, спектральный анализ.
MAC – Multiplication with accumulation.
аналоговый сигнал (x(t)) → АЦП(дискретный сигнал X(nT)) → ЦПС(дискретный сигнал Y(nT)) → ЦАП = (Y(t), аналоговый сигнал)

7.1 Фильтр скользящего среднего

$$Y(nT) = \frac{1}{3}\sum X(nT) + x(nT - T) + X(nT - 2T)$$

$$Y(nT) = 0.33\sum_{i=0}^{2} X(nT - iT)$$

Частота сигнала F_c
Частота дискретизации $F_д \geq 2F_c$
Например:

$$F_c = 100MHz; F_д = 200MHz; T = 5нс$$

7.2 Основные классы DSP

- DSP с фиксированной точкой. Чаще всего 16 разрядов
- DSP с фиксированной точкой (расширенной точности), 24 разряда
- DSP с плавающей точкой 32 бит.

Основные производители:

- Texas Instruments 40%
- Freescale Semiconductors 25%
- Analog Devices 10%

7.3 Формат данных

S – знака	24 бита	S(23)	дробная часть(22-0)	
	48 бит	S(47)	дробная часть(46-0)	
	56 бит	S(55)	целая часть (54-47)	дробная часть(46-0)

7.4 Структура DSP

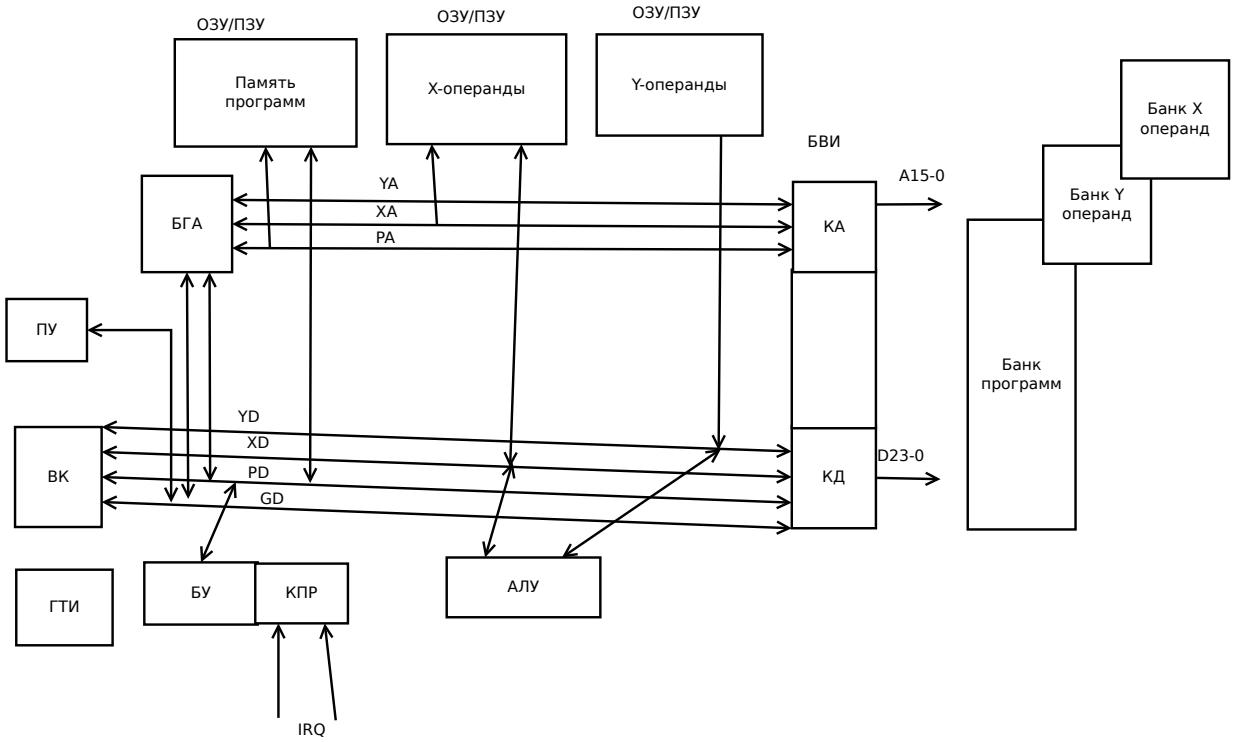


Рис. 7.1: Структура DSP

- БГА – Блок генерации адрессами. Может работать одновременно с тремя адрессами: считывание адреса и операндов.
- ПУ – переферийные устройства
- ВК – внутренние коммутаторы (пересылка, подключая хвосты шины)
- Ка – коммутатор адреса
- КД – коммутатор данных
- БУ – блок управления. Дешифрирует комманды, выдает сигналы для управления.
- КПР – контроллер прерываний. Воспринимает внешние прерывания и обеспечивает их обработку.
- АЛУ – Арифетическо-логические операции
- БВИ – блок внешнего интерфейса
- ГТИ – генератор тактовых импульсов. Внутренний.
- XA, YA, PA – шины адреса
- XD, YD, PD, GD – шины данных.

Такую структуру называют иногда двойной гарвардской.

Имеется внутренняя память программ данных. Есть внешняя, которую можно использовать. Внутренняя память зависит от модели 0.5-8КБ. Часть память – ОЗУ, часть – ПЗУ.

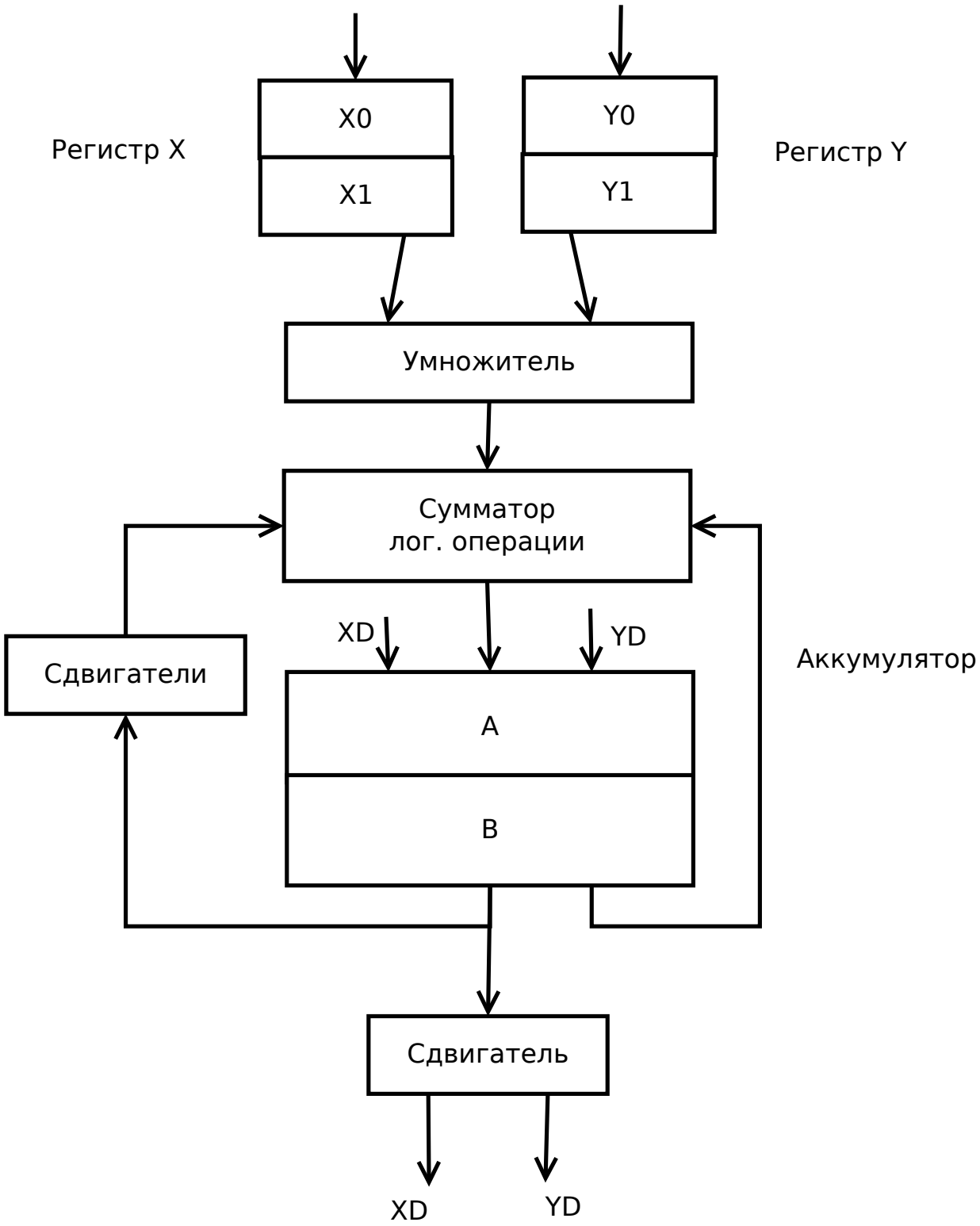


Рис. 7.2: АЛУ DSP

x_0, x_1 (24 разряда) можно объединить в один регистр X (48 разрядов)
A, B – 56 бит аккумуляторы.

A2(B2)	A1(B1)	A0(B0)

MAC-операция выполняется за один такт.

7.5 Регистровая модель

7.5.1 Регистры АЛУ

X1(23-0)	X0(23-0)			X	Y1	Y0	Y
000	A2	A1	A0	A			
0 0	B2	B1	B0	B			

7.5.2 Регистры Адреса

R0 (15-0)	N0 (15-0)	M0 (15-0)
...	...	...
R7	N7	M7

- R – регистры адреса
- N – регистры смещения
- M – регистры модификации
- Косвенно-регистровая модель (Ri)
- Косвенно-регистровая со смещением (Ri+Ni)

Регистр M

Организация в памяти циклических буферов. Выборка коэффициентов (записали что-то, а потмо вернулись чтобы выбрать коэффициенты заново).

7.5.3 Регистры управления

LA (15-0)
LC

OMR (15-0)

OMR – регистр режима

PC(15-0)	SR)15-0

SP – аппаратный стек. SS0-SS15
LA – адрес конца цикла.
LC – счетчик циклов
PC – программный счетчик.
Приложения:

- C –
- N – знак
-

Управляющие биты:

- I – рез

7.6 Способы адресации

1. Регистровая. Указывается имя соответствующего регистра:
   X,Y,A,B – АЛУ. Rn, Nn, Mn – адрес, LC, LA, OMR, SR – управляющие. См. 7.5
2. Косвенно-регистровая. (Rn)
3. Косвенно-регистровая с пост-инкрементом (Rn) +1
4. Косвенно-регистровая с пост-инкрементом на N – (Rn)+(Nn)
5. Косвенно-регистровая с пост-декрементом – (Rn)-1
6. Косвенно-регистровая с пост-декрементом на N – (Rn)-(Nn)

7. Косвенно-регистровая с пред-декрементом – (Rn-1)
8. Косвенно-регистровая со смещением – (Rn+Nn)
9. Прямая (адрес в команде)
10. Непосредственная (операнд в команде)

7.7 Система команд (особенности)

7.7.1 Арифметические операции

- ADD s, d

  s: X0, X1, Y0, X2; d – A или B; s – source, d – destination

- ADC (ADD + перенос)
- SUB (вычитание)
- SBC (вычитание с переносом)

Особенности состоят в том, что арифметические операции АЛУ могут совмещаться с операциями загрузки: ADD x0, A, X:(R0), X0, Y:(R2),Y0
Из X-ОЗУ в X0, из Y-ОЗУ в Y0.
Это представляет из себя элемент VLIW-архитектуры. Это возможности благодаря множеству линий (7.2)
Можно выполнять одновременно сложение со сдвигом:

- Сдвиг влево: ADDL S,D – S+D*2→D
- Сдвиг вправо: ADDR S,D – S+D/2→D

Операция умножения с накоплением:

- MPY s1,s2,D – S1*S2 → D
- MAC s1,s2,D – S1*S2 + D →D

Вычитание со сдвигом: SUBL, SUBR
DIV S, D D/S → 1 разряд частного (младший).
Чтобы получить полный результат деления нужно использовать операцию повторения:

- REP #n – повторение предыдущей команды n раз.
- REP s – n в регистре X или Y
- REP x(Rn) – n в X или Y памяти

Логические операции:

- ADD s,D – s → x0, x1, y0, y1; D = A1 или B1
- OR
- EOR
- NOT
- ANDI #n, Rc – Rc – регистр управления. Используется для модификации регистров управления. Цель этих операций – изменение регистров LA, LC, OMR, SR.
- ORI #n, Rc

  Выбирается 24 разряда для логических операций.

Сдвиг (1 разряд)

- ASL D – D – A или B. (арифметический)
- ASR

- LSL (логический)
- LSR
- ROL (циклический)
- ROR

  Выдвигающийся бит – C

Команда пересылки:

- MOVE S,D (пересылка АЛУ)
- MOVEC x:addr, Ri – регистр управления

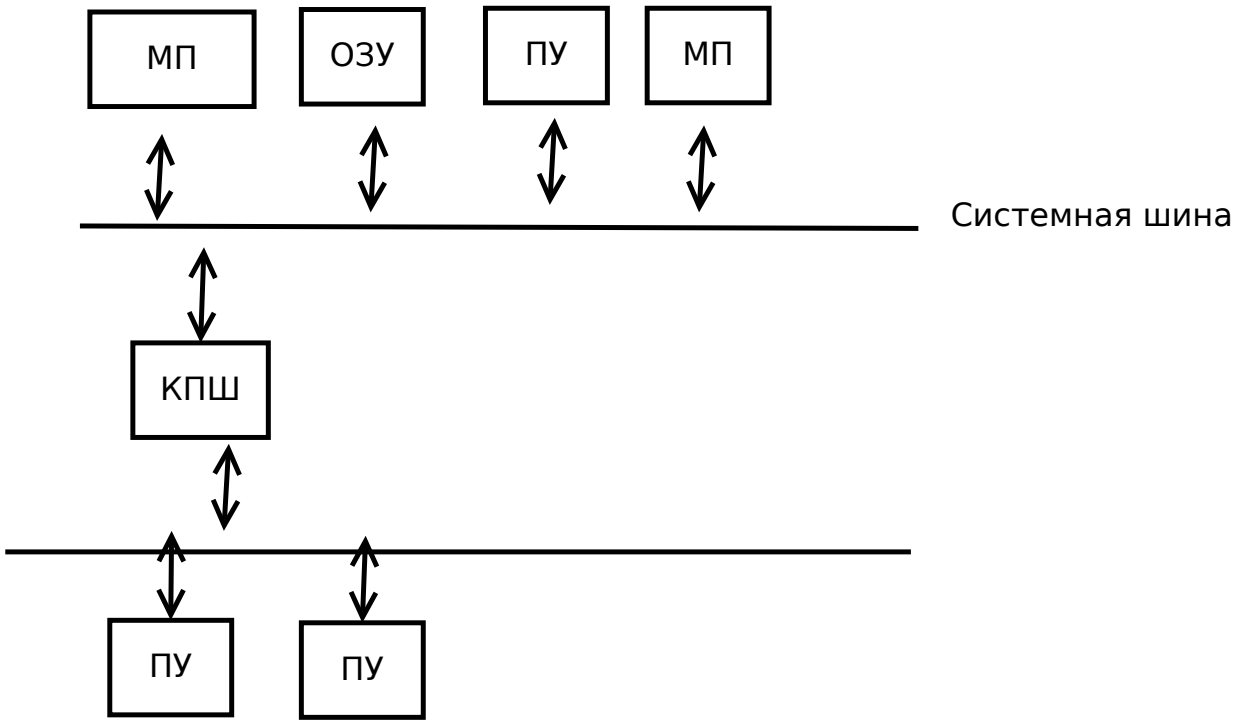
Битовые операции:

- BTST #n, X:addr – вместо X мб Y. n – номер бита (0-23). bn → C
- BSET – bn→c, 1→bn
- BCLR – bn→c, 0→bn
- BCHG – bn→c; !bn→bn

Команды управления
Безусловный переход JMP addr ; addr→PC
Условный: JCC addr – addr→PC
JCLR #n, addr1, addr2 ; addr1 – адрес операнда, addr2 - перехода, переход на addr2 если bn=0
JSET – bn=1
JSR addr – PC → SP, addr→PC.
RTS – возврат из JSR. апп. стек → PC
Програмные прерывания:
SWI – PC, SR→ SP. Вектор → PC
Возврат – RTI – SP → PC=SR
Оргранизация циклов:
DO #n, addr1 – n - число циклов (LC), addr – адрес окончания (LA) (адрес команды выхода из цикла)
LC-1 после каждого цикла. ENDDO – выход до окончания цикла
DO R, addr – число циклов в X, Y
DO x:addr1, addr2 – addr – адресс ячейки в ОЗУ (X или Y)

Глава 8

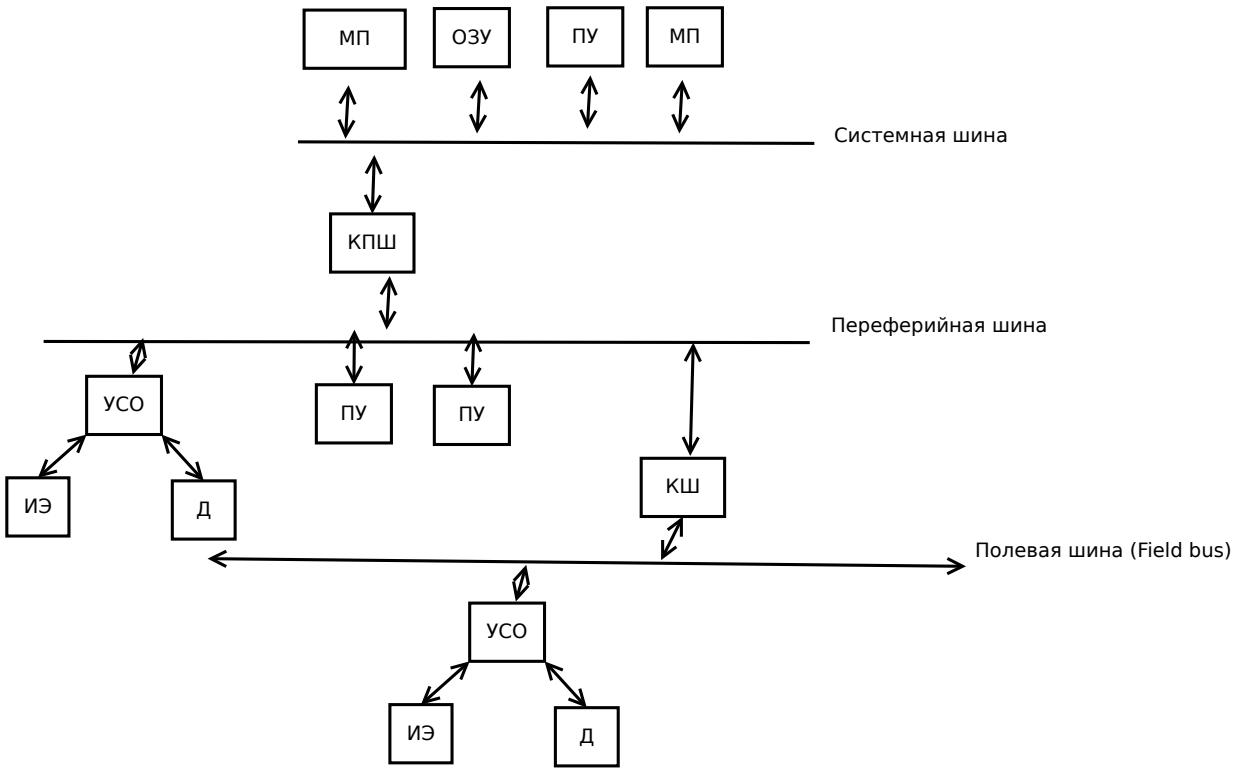
Организация интерфейса в системах



Неудобно когда переферийные устройства сидят на системной шине, поэтому используют специальную переферийную шину.

КПШ – контроллер переферийной шины

Разновидностью ПУ можно считать УСО (Устройство связи с объектами, Д – датчики, ИЭ – Измерительные элементы).



Вводят специальную шину (полевую шину) для связи с переферийными устройствами

8.1 Интерфейс

Что же такое интерфейс?

Интерфейс – совокупность аппаратных, программных средств и конструктивов, обеспечивающих корректный обмен данными.

Составные части интерфейса:

- Правила обмена (протоколом).
- Аппаратная реализация – физический уровень.
- Программное обеспечение.

8.1.1 Шины.

Типы шин

- параллельные (ISA, PCI, VME)
- последовательные (PCI-Express, USB, CAN, I2C)
- С квитированием (специальные сигналы запроса и подтверждения)
- Без квитирования
- С мультиплексированием (по одной линии идет Addr и затем Data) [Только параллельные]
- Без мультиплексирования (A и D идут по разными линиям) [Только параллельные]
- Дуплексные (одновременная передача и прием сообщений. У1 и У2 имеют как Rx, так и Tx выходы) [Только последовательные]
- Полудуплексные (одна линия и для приема и для передачи)

Арбитраж шин

Арбитраж по-разному организуется для последовательных и параллельных шин.

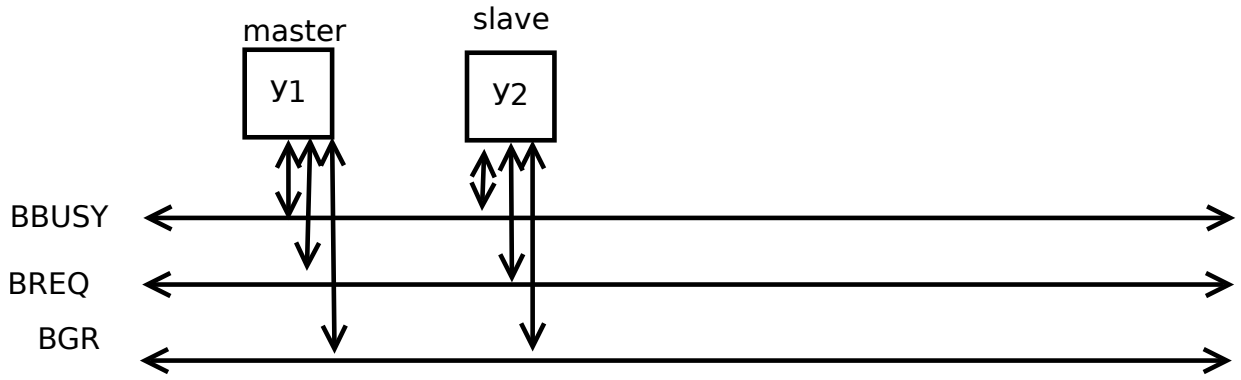


Рис. 8.1: Арбитраж

BBUSY – захват. BREQ – запрос. BGR – подтверждение

В начале BBUSY, захвад шины ведущим. В момент времени 1 приходит запрос BRQ, через некоторое время мастер отдает шину ведомому генерируя сигнал подтверждения. Ведомый снимаает запрос.

8.1.2 Представление данных в последовательных шинах

Используются различные варианты кодировки данных:

Код	описание представления на примере последовательности 110010	комментарий
NRZ	110010	1 - это 1, 0 - 0
NRZI	110110	1 сохранение знака, 0 - изменение. Например USB
FM	010100001011	Фазомоделированный. В середине такте происходит переключение, если 1
Манчестерский	101001011001	При 1 - переключение из 1 в 0. При нуле – из нуля в единицу. (на полу-такте)

Достоинства манчестерского кода – на каждом такте происходит переключение. Принимающее устройство знает частоту и может под нее подстроиться.

8.1.3 Последовательный обмен по стандарту RS232

Для понимания можно воспользоваться ссылкой: `http://www.gaw.ru/html.cgi/txt/interface/rs232`

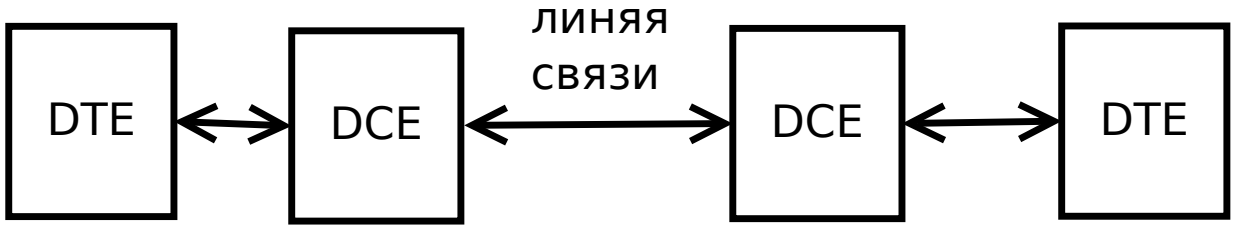


Рис. 8.2: RS232

DTE (Data Terminal Equipment) – колтитер (компьютер?)
DCE (Data communication Equipment) – модем
Разъем 9-и контактный
TxD – линия передачи

RxD – линия приема
Линия квитированная.
Сигналы квитирования:
DTR – Data Transmit Request – запрос DTE
DSR – готовность модема.
(справа на рисунке DTE → RTS; DSR → CTS)
RTS – готовность терминала. CTS – готовность модема.

8.2 Сетевой интерфейс CAN

RS232 не позволяет организовать сеть. Он дает только соединение одного источника. Для сетевых используется несколько более сложный протокол CAN.

С помощью одного провода.

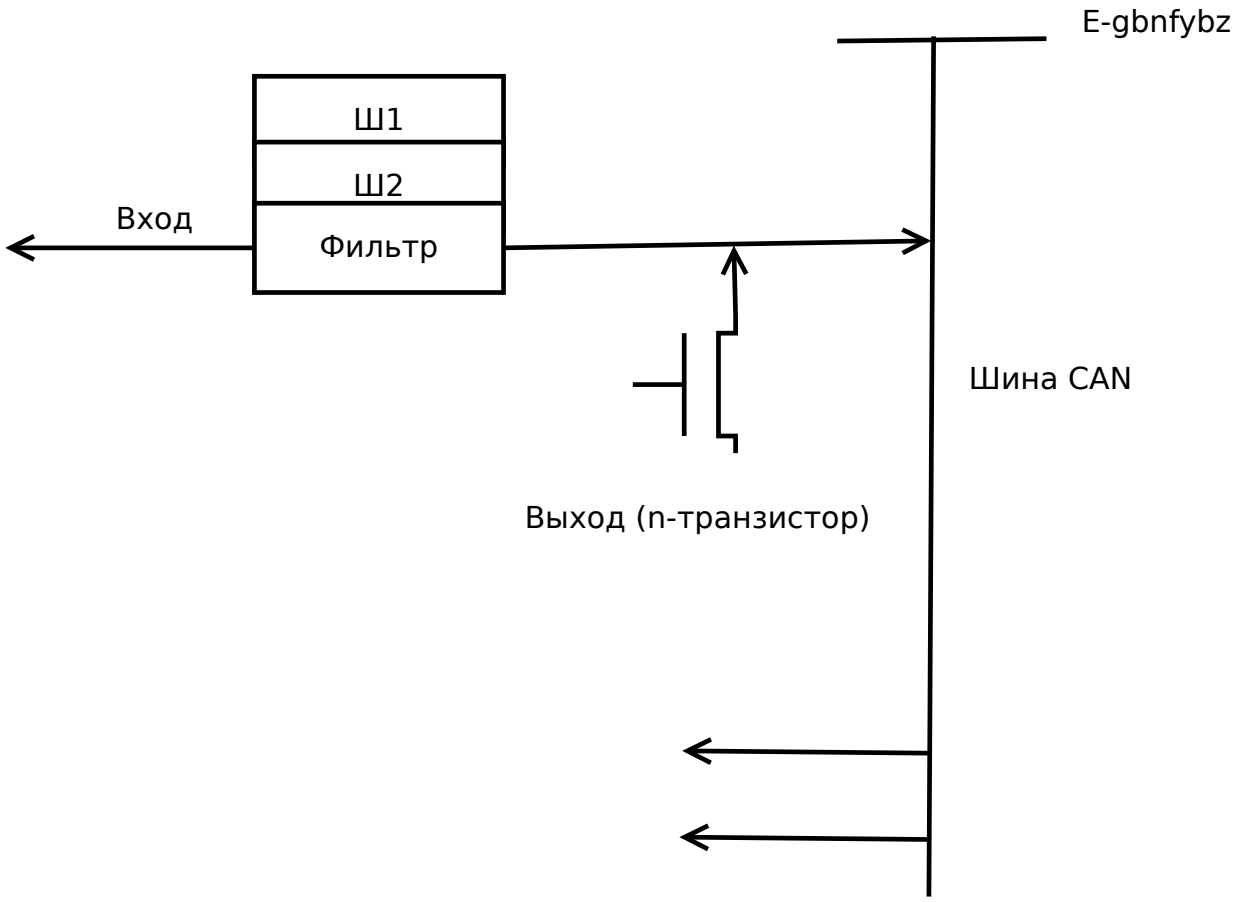


Рис. 8.3: CAN

- Выход с отккрытым стоком.
- Каждое сообщение имеет идентификатор ID. ID код 11 или 29 бит. 0 – доминантный бит, 1 – рессивный бит.

  11 бит – 2048 сообщений, 29 бит – 256 миллионов.
- Каждый контроллер настроен на определенное число идентификаторов (шаблонов)
- Контроль ошибок при передаче
- Передача идет кадрами (Frame). Каждый кадр имеет фиксированный формат:

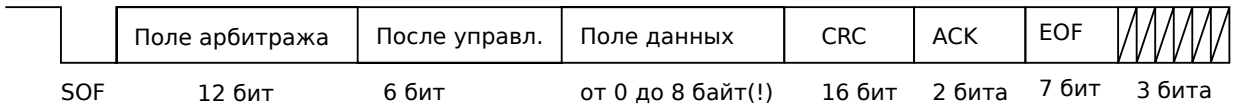


Рис. 8.4: Кадр

ID в поле арбитража. SOF – Start of frame. CRC – контрольная сумма. EOF – 7 рецессивных бит.

Поле арбитража содержит ID (11 бит), затем 1 бит RTR – тип сообщения, данные(1) или запрос(0).

Поле управления:

IRE(1 бит)	R0 (1)	D2C (4)

IRE – 0 – стандартная форма. 1 – расширенная

r0 – резервный бит 0

D2C – размер данных (от 0 до 8 байт)

Поле ACK – передача – 2 рецессивных. Прием – 2 доминантных бита.

8.2.1 Ошибки синхронизаций

Допустим разброс в частоте 10%, тогда при большой последовательности чисел будет теряться каждый 10 бит.

Чтобы это избежать делается бит-стаффер, вставка бита. После пяти одинаковых ставится один противоположный.

Приемник делает обратную процедуру – если принял пять бит, то шестой удаляем.

Таким образом обеспечивается самосинхронизация.

8.2.2 Контроль ошибок

Типы ошибок:

- Сравнение CRC на стороне приемника.
- Ошибка формата – R0 = 1, EOF<7 и т.п.
- Ошибка ACK: приемник или передатчик
- Ошибка передачи (не получен ACK?)
- Ошибка битстаффера.

В случаи обнаружения, контролер CAN формирует кадр ошибки – 6 бит. В ответ другие контроллеры устанавливают ещё 6 бит. Итого 12 доминантных бит (в скобках указано количество бит):

Форма ошибки (6)	Дополнительные флаги (6)	разделитель (8)	промежуток (3)

Получив кадр ошибки, передатчик повторяет сообщение.

Каждый контроллер в сети имеет два счетчика: счетчик ошибок приема и счетчик ошибок передачи. Обычно это 8-и разрядный счетчик с максимумом 255 ошибок.

Если 127 ошибок – перевод в пассивный режим (может передавать только свои сообщения). Если счетчик переполнился, то контроллер отключается. Это нужно, потому что если контроллер неисправен, то он может дублировать свои ошибки и потенциально может заблокировать сеть.

Для сброса контроллера есть сигнал reset.

8.2.3 Особенности

- Требует только одну линию.
- Эффективный контроль ошибок. Выявляет и исправляет целый ряд ошибок. Неисправные контроллеры отключаются.
- Вместо адреса используются ID. Это удобно по ряду причин. Не нужно менять адрессное устройство при добавлении нового устройства (устройство со своими шаблонами принимает то, что нужно).
- Легко реализовать широковещательный режим.
- Разрешение конфликтов. Приоритет имеет сообщение с меньшим ID. Идет совподающий ID, K2 имеет доминантный бит, K1 прервет передачу.

8.2.4 Сфера применения

- Автомобильная электроника.
- Промышленая электроника
- Контрольно-измерительные системы.

Приложение А

Архитектура SPARC. Серия лекций МЦСТ

А.1 Архитектура SPARC-2

- RISC-архитектура.
- 3 ревизии:
  - V7, V8: 32-х битная.
  - V9: 64-х битная.
- Чипы делают:
  - SUN
  - Fujitsu
  - Gaisler Research
  - ЗАО «МЦСТ ».
- Универсальные микропроцессоры для целочисленных и плавающих вычислений
- Лицензионная и патентная чистота. Лицензия стоит дешево (100$ за любое количество процессоров)
- Полная аппаратная совместимость с архитектурой SPARC V8 (sunv4, планируются переход на v9, т.к. v8 устарел лет 10 назад):
  - RISC-архитекутра.
  - 3-адресная команда (в каждой команде закодировано 3 регистра)
  - формат данных: целочисленные 32 разряда. Вещественные 32, 64, 128 (как правило не делают)
  - Многооконный регистровый файл
- Возможность использования большого массива стороннего программного обеспечения.

А.2 Особенности SPARC-машины

1. Счетчики команд:
   - PC – указатель текущей команды
   - nPC – указатель следующей команды

   Оба счетчика всегда видны. В SPARC переход изменяет nPC – в классической RISC-машины с 5-и стадийным конвейером нужно успеть декодировать команды, поэтому так схитрили, чтобы не терять такт (задержанный переход – nPC = target_VA)
2. Состояние процессора:
   - V8: PSR (processor state register)

- PSTATE

3. Команда, следующая за переходом может быть анулирована
4. RISC-архитектура: память доступна только при помощи команд LOAD и STORE. Память может только прочесть и записать.
5. Только два режима адрессации:
   - Регистр + регистр
   - Регистр + непосредственный операнд (13 бит со знаком)

   На SPARC стараются избегать Assembler'a.
6. Синхронизация:
   - V8: LDSTUB – неразрывный обмен. Если ОС исполняется одновременно на двух ядрах и имеет доступ к данным, их нужно не повредить. Все умные вещи типа spinlock'ов, барьеров и т.п., строятся на примитивах типа LDSTUB (LOAD/STORE UNSIGNED BYTE).
   - V9: CAS – неразрывное сравнение и обмен, LDSTUB – неразрыный обмен.
7. V9: есть программно управляемеая предвыборка (PREFETCH). Можно префетчить в любой из кэшей, можно префетчить инструкции. Например Branch Never – префетч инструкции в кэш.
8. Интерпретация VA (virtual address) задается ASI (Address Space Identifier). У каждого оператора есть параметр ASI, задающий тип интерпретации. Преобразование VA при помощи страничной адресации. Для LOAD/STORE ASI вычисляется неявно. В V9 можно таким образом считать сразу несколько регистров.
9. Адрес регистра в команде – 5 разрядов.
10. Ргеистры делятся на 4 вида:
    - Глобальные/global (r0-r7)
    - Выходные/output (r8-r15)
    - Локальные/local (r16-r23)
    - Входные/input (r24-r31)
11. Регистр CWP (Current Window Pointer) указывает текущее окно. Этот регистр указывает место в регистровом файле на конкретную группу регистров (регистровый файл может содержать порядка 160 регистров)
12. Полный адрес регистра образуется путем конкатенации регистра CWP с младшими разрядами номера регистра (на самом деле сложнее)
13. Выходные регистры окна с номером N+1 совпадают с входными из окна с номером N: окна перекрываются.
14. Для изменения CWP используются инструкции SAVE (CWP–) и RESTORE(CWP++). Вызов процедуры не меняет регистр CWP. Можно сделать CWP++ и выходные регистры станут входными. Это позволяет проще переиспользовать код (вызов процедур). Если работы мало, то процедура может работать прямо в имеющихся регистрах. При этом процедура может поменять окно, если ей нужно.
15. Регистры %g0 всегда равен нулю.
16. Для управления окнами используется регистр WIM (). Если Save или RESTORE наталкиваются на окно, замаскированное в регистре WIM, происходит прерывание.
17. Одно окно всегда зарезервировано для обработки прерываний.
18. В версии V9 механика управления окнами переработана (нет WIM, но для пользовательских программ не заметно, т.к. они с регистрами напрямую не оперируют)
19. Плавающие регистры организованы плоско (flat register file).
20. Прерывание: передача управления операционной системе через таблицу прерываний (trap table)
21. Таблицы обработчиков: TBR={TBA,TT}
22. Виды прерываний (как в V8):

- Точное: команда, на которой произошло прерывание не успела изменить состояние процессора, для восстановления состояния не надо прилагать усилий
- Отложенное: изменения состояния были, но его можно восстановить
- Разрушающее: точная привязка к команде невозможна.

23. Вложенные прерывания:
    - V8: фатальная ошибка
    - V9: 5 уровней вложенности
24. Причины прерываний:
    - Исключительная ситуация, связанная с выполняющейся перед этим командой
    - Исключительная ситуация, НЕ связанная с выполняющейся перед этим командой
    - Внешнее прерывание
25. Внешние прерывания: управляются регистром PIL и битом IE

A.3 Микропроцессоры MCST-R 2

- 4 поколения микропроцессоров архитектуры SPARC V8
- Процессы 0.5-0.13мкм
- Частота: 80-500МГц.
- Последний микропроцессор с архитектурой V8: MCST R500S:
  - Двухъядерная система на кристалле
  - Пятистадийный классический RISC-конвейер
  - Технология изготовления: TSMC 0.13um LVLK
  - Частота 500МГц
- Задачи при разработке микропроцессора следующего поколения:
  - Миграция на архитектуру V9
  - Суперскалярная микроархитектура
  - Частота 1ГГц
  - Поддержка ccNUMA

A.4 MCST-4R: система на кристалле

- 4 процессорных ядра на чипе
- 2МБ общий кэш L2
- Интегрированный контроллер DDR2
- Объединение до 4-х чипов в ccNUMA без дополнительной логике
- Южный мост – отдельный чип (также разработан в МЦСТ)
- Доступ в южный мост по LVDS-линку.

A.4.1 Ядро

- Набор инструкций: SPARC V9
- SIMD расширения VIS
- Операция сложения с умножением
- 7-тактный целочисленный конвейер
- Суперскаларное выполнение инструкций
- Статическое планирование. Компилятор может предсказать выполнение операций на конвейере.
- Операции IU0:
  - Сложнение, сдвиг, логические
  - Генератор адреса обращения в память
- Операции IU1:
  - Сложение, сдвиг, логические
  - Умножение
  - Деление
- Суперскалярный конвейер: 2 инструкции за такт
- Возможные сочетания команд:
  - int + int
  - itn + mem
  - int + fp
  - int + control transfer
- 64-х разрядный виртуальнй адрес, 40 разрядный физический
- Все кэш-памяти первого уровня поддерживают доступ со стороны процессора И снупинг каждый такт
- L2 кэш поддерживает доступ со стороны процессора ИЛИ снупинг каждый такт
- Протокол когерентности кэш-памятей MOESI в L1 данных и L2
- Шина данных между ядром и L2 шириной 256 бита
- TLB программно наполняемы
- L1 кэш данных с отложенной записью снижает требования к L2 по полосе пропускания.
- Характеристика кэш-памятей:

Cache	L1I	L1D	L2
Size	16K	32K	2M
Block size	64	64	64
Protection	Parity	Parity	ECC
Policy	-	WB	WB
Associativity	2	4	32

- Для достижения необходимых частот сделаны двухуровневые TLB

A.4.2 ccNUMA

- В МЦСТ разработан ccNUMA протокол.
- 3-х хоповый, на основе широковещательны сообщений
- Контроллер когеренотности обнаруживает и разрешает конфликты
- Системные линки имеют ширину 16 бит пропускная способность 1GT
- Линк ввода-вывода может быть использован для создания кластера в режиме RDMA.

A.4.3 Физический дизайн

- Мрашрут проектирования: Standard Cell'ы, тулы Synopsys
- Заказной маргшрут: целочисленный регистровый файл
  - 5 портов чтения (4 целочисленых операнда и 1 слово данных на заирись)
  - 3 портов записи (2 целочисленных результата и 1 результат из памяти)
- Целевая частота 1ГГц
  - Разбалансировака дерева распространения синхросигнала
  - RTL оптимизируется для достижения частоты
  - Перемещение логики между ступенями конвейера (например предварительно декодированный код вкэш памяти инструкций)
- Рассеиваемая мощность
  - Автоматическое отключение синхросигнала на неработающей логике
  - Ручное отключение синхросингала на неработающей логике (например в FPU)
  - Использование транзисторов с низким/средним/высокими порогом.
- Синтез частоты для зразличных доменов синхронизации
  - Deterministic fractional ratio clocking for DDR2
  - Other domains have integer clock ratio
- На приемнике LVDS линка нужен DLL – DLL построен на standard cell'ax, охарактеризован в SPICE
- Площадь ядра: 7.6мм^2
- Площадь кристалла: $\sim$115мм^2

Приложение B

Архитектура VLIW и микропроцессоры Эльбрус

Т.к. данная глава является презентацией, то конспектироваться будет только то, что в ней отсутствовало.

B.1 1 слайд

Логическая скорость вычисления – то, что действительно может дать микроархитектура. IPC и CPI – термины, встречающиеся в литературе.

B.2 2 слайд

Stall может возникать потому что у нас просто нечего исполнять, может быть неготовность операндов (какое-нибудь умножение или обращение в память) и так далее.

IPCmax – для одноконвеерной машине = 1. Пиковая производительность.

При распараллеливании IPCmax будет равен «ширине»аппаратуры. Появляется коэффициент утилизации (k_util).

При проектировании стараются бороться с k_stall и увеличить k_util.

B.3 4 слайд

Программист пишет исходную программу. Строится граф зависимостей программы. В суперскалярных архитектурах код тоже последовательный, но исполняется частично параллельно (на уровне аппаратуры идет перестановка операций и так далее). Подход VLIW – такой же эффект достигается компилятором.

B.4 5 слайд

EPIC – Explicity Parallel Instruction Computing.

B.5 6 слайд: Выбор архитектуры VLIW

Возникла архитектура на всевозможных DSP. В процессорах общего назначения она почти не использовалась из-за сложностей.

B.6 Характеристики процессоров

Эльбрус – 2006-2007 г., но спроектирован он был в начале века. Частота 300МГц. Было довольно мало ошибок, около трех итераций прежде чем вышел приличным. Производительность – на разных задачах Pentium 4 600MHz до 10GHz (пересчет). Техпроцесс - 0.13мкм. Потребление - 6-9Вт. FLOPs: 4 вещественных умножить на частоту (6 целочисленных). Есть SIMD-операции для 32-х разрядных чисел - они вдвое быстрее.

Эльбрус-С – разрабатывается. Частота 500МГц, но только прошла первая итерация, поэтому конкретные характеристики неизвестны. Технология - 90нм. Потребление порядка 11Вт. Изготовлен на TSMC.

B.7 Применение

VLIW малораспространен, т.к. они хорошо подходят для рассчетов – их очень тяжело эффективно использовать. Управляющая программа содержит огромное количество ветвлений и её критический путь проходит фактически по всему дереву, её быстрой сделать просто невозможно. Как следствие – статическое управление сильно проигрывает динамическим машинам.

Итог: VLIW не является сильно универсальной.

С другой стороны – вычислительная техника начиналась с одноконвеерных машин. x86 захватил рынок и нужно поддерживать совместимость со старым ПО.

B.8 Область применения

Архитектура применяется в DSP, но в более упрощенном виде.

Область применения Эльбруса – имеется спец. заказчик (военные).

Цена – зависит от партии. При партии 1000 штук – 1500$. Одна итерация стоит фиксированно – 1.5кк$, а количество чипов на цену почти не влияет.

B.9 Особенности Эльбруса

Изначально закладывались средства для исполнения кода x86 через динамическую перекомпиляцию (DynComp).

Приложение C

Проведение экзамена

Достаточно знать пару команд ассемблерных из каждой группы.
Пересдача 25.01 и 01.02. Возможно устроить дополнительные консультации.
Шпоры будут палить и за них удалять.
Если билет не понравится, можно взять второй, но будет -1 к оценке.
Автоматы есть, но какие-то странные и непредсказуемые. Лучше на них не рассчитывать.

Приложение D

Вопросы к экзамену

1. Обработка числе с плавающей точкой в микропроцессорах Пентиум. Форматы обрабатываемых чисел. Регистровая модель блока FPU, назначение основных регистров.
2. Обработка чисел с плавающей точкой в микропроцессорах Пентиум. Набор команд, выполняемых FPU, используемые способы адресации.
3. Групповая обработка целочисленных операндов в МП Пентиум. Форматы обрабатываемых данных. Регистровая модель блока MMX. Набор команд, выполняемых MMX, используемые способы адресации.
4. Групповая обработка операндов с плавающей точкой в микропроцессорах Пентиум. Форматы обрабатываемых данных. Регистровая модель блока SSE. Наборы команд, выполняемых SSE, используемые способы адресации.
5. Особенности архитектуры современных RISC-процессоров (на примере MIPS). Структура и функционирование 32-разрядных RISC-процессоров семейства MIPS. Регистровая модель центрального процессора CPU.
6. Регистровая модель сопроцессора управления CP0 в микропроцессорах семейства MIPS. Назначение и функционирование сопроцессора управления.
7. Способы адресации и система команд RISC-процессоров семейства MIPS.
8. Построение систем на базе микропроцессоров MIPS с использованием системного контроллера. Реализация параллельного обмена данными, таймерных функций, векторных прерываний.
9. Особенности архитектуры современных RISC-процессоров (на примере PowerPC). Регистровая модель RISC-процессоров семейства PowerPC. Формат команд и способы адресации.
10. Система команд RISC-процессоров семейства PowerPC.
11. Особенности архитектуры современных RISC-процессоров (на примере ARM). Регистровая модель RISC-процессоров семейства ARM. Формат команд и способы адресации.
12. Система команд RISC-процессоров семейства ARM.
13. Структура и функционирование цифровых процессоров сигналов DSP56xxx.
14. Регистровая модель цифровых процессоров сигналов DSP56xx, реализуемые способы адресации.
15. Система команд цифровых процессоров сигналов DSP56xxx.
16. ОСРВ ОС2000. Архитектура ОС2000, временные характеристики. Потоки управления, сигналы, средства синхронизации в ОС2000.
17. Основные службы ядра ОСРВ. Управление задачами, взаимодействие процессов, динамическое распределение памяти.
18. Последовательные интерфейсы, способы кодирования двоичных данных. Интерфейс стандарта RS-232.
19. Организация передачи данных по последовательному интерфейсу CAN.

Приложение E

Билеты

Билет №1
1. Обработка числе с плавающей точкой в микропроцессорах Пентиум. Форматы обрабатываемых чисел. Регистровая модель блока FPU, назначение основных регистров.
2. Особенности архитектуры современных RISC-процессоров (на примере ARM). Регистровая модель RISC-процессоров семейства ARM. Формат команд и способы адресации.

Билет №2
1. Обработка чисел с плавающей точкой в микропроцессорах Пентиум. Набор команд, выполняемых FPU, используемые способы адресации.
2. Система команд RISC-процессоров семейства ARM.

Билет №3
1. Групповая обработка целочисленных операндов в МП Пентиум. Форматы обрабатываемых данных. Регистровая модель блока MMX. Набор команд, выполняемых MMX, используемые способы адресации.
2. Структура и функционирование цифровых процессоров сигналов DSP56xxx.

Билет №4
1. Групповая обработка операндов с плавающей точкой в микропроцессорах Пентиум. Форматы обрабатываемых данных. Регистровая модель блока SSE. Наборы команд, выполняемых SSE, используемые способы адресации.
2. Регистровая модель цифровых процессоров сигналов DSP56xx, реализуемые способы адресации.

Билет №5
1. Особенности архитектуры современных RISC-процессоров (на примере MIPS). Структура и функционирование 32-разрядных RISC-процессоров семейства MIPS. Регистровая модель центрального процессора CPU.
2. Система команд цифровых процессоров сигналов DSP56xxx.

Билет №6
1. Регистровая модель сопроцессора управления CP0 в микропроцессорах семейства MIPS. Назначение и функционирование сопроцессора управления.
2. ОСРВ ОС2000. Архитектура ОС2000, временные характеристики. Потоки управления, сигналы, средства синхронизации в ОС2000.

Билет №7
1. Способы адресации и система команд RISC-процессоров семейства MIPS.
2. Основные службы ядра ОСРВ. Управление задачами, взаимодействие процессов, динамическое распределение памяти.

Билет №8
1. Построение систем на базе микропроцессоров MIPS с использованием системного контроллера. Реализация параллельного обмена данными, таймерных функций, векторных прерываний.
2. Последовательные интерфейсы, способы кодирования двоичных данных. Интерфейс стандарта RS-232.

Билет №9
1. Особенности архитектуры современных RISC-процессоров (на примере PowerPC). Регистровая модель RISC-процессоров семейства PowerPC. Формат команд и способы адресации.
2. Организация передачи данных по последовательному интерфейсу CAN.

Билет №10
1. Обработка числе с плавающей точкой в микропроцессорах Пентиум. Форматы обрабатываемых чисел. Регистровая модель блока FPU, назначение основных регистров.
2. Система команд RISC-процессоров семейства PowerPC.