Архитектура ЭВМ и основы ОС

Система команд ARM

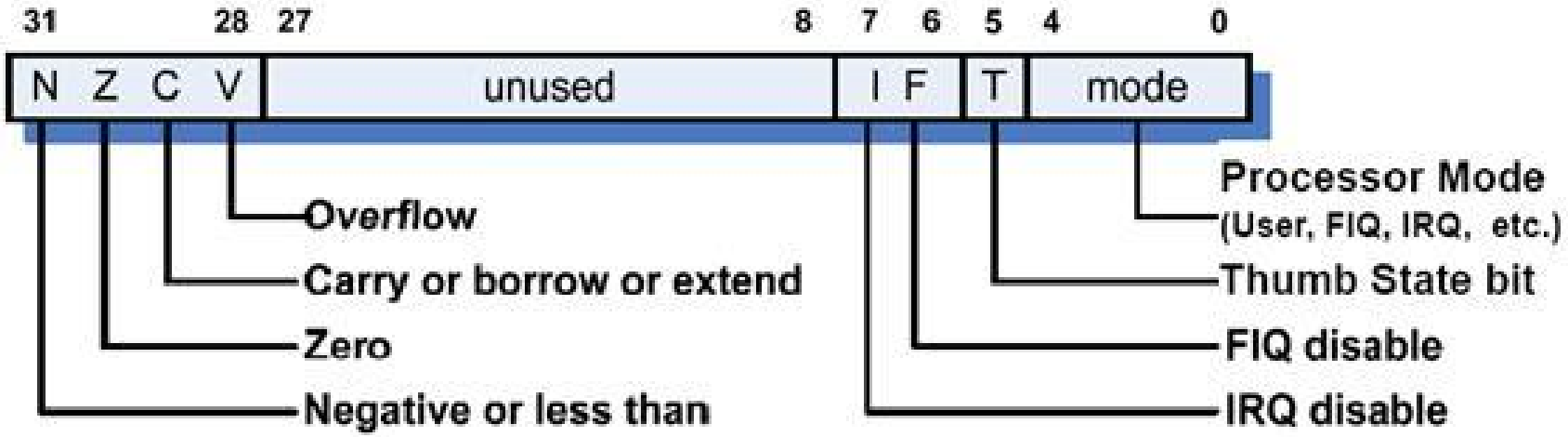
RISC

- RISC - redused instruction set computer:
  - минимальное количество реально используемых команд
  - большое количество регистров
  - почти все операции выполняются над регистрами
  - команды фиксированной длины и “сложности”

История ARM

- ARM1 (1985)
- ARM2 (1986) - первая серийная модель:
  - 32-битная шина данных
  - 26-битное адресное пространство
  - 16 32-битный регистров
  - ~ 30000 транзисторов (против ~70000 у Motorola 68000)
- ARM6 (1992)
- ARM7 (1993) - добавляются наборы команд Thumb, переход к 32-битной адресации
- ARM11 - добавляется SIMD и Thumb-2
- Cortex M (контроллеры), Cortex R (real-time), Cortex A (приложения)
- Cortex A50

CPSR и SPSR



Program status register format

- ARM CPSR format -

Регистры

- Регистры общего назначения R0-R12
- Указатель стека SP или R13
- Регистр связи LR или R14 (содержит адрес возврата)
- Счетчик команд PC или R15

- Все регистры кроме PC и R0-R7 дублируются, а всего может быть 31-33 регистров.

Пример

```
    .text
    .global  dummy
    .type    dummy, %function
dummy:
    bx   lr
    .size    dummy, .-dummy
```

- bx{COND} Rn
  - Rn - регистр, содержащий адрес перехода
  - bx - переход и выбор режима процессора (ARM или Thumb, в зависимости от 0 бита адреса)
  - {COND} - опциональное условие выполнения (например, LS)

Коды условий

Код	Условие	Тип	Значение
EQ	Z == 1	знаковый и беззнаковый	Res == 0
NE	Z == 0	знаковый и беззнаковый	Res != 0
CS/HS	C == 1	беззнаковый	Op1 >= Op2
CC/LO	C == 0	беззнаковый	Op1 < Op2
HI	C == 1 && Z == 0	беззнаковый	Op1 > Op2
LS	C == 0 || Z == 1	беззнаковый	Op1 <= Op2

Коды условий

Код	Условие	Тип	Значение
MI	N == 1	знаковый	Res < 0
PL	N == 0	знаковый	Res >= 0
VS	V == 1	знаковый	переполнение
VC	V == 0	знаковый	без переполнения
GE	N == V	знаковый	Op1 >= Op2
LT	N != V	знаковый	Op1 < Op2
GT	Z == 0 && N == V	знаковый	Op1 > Op2
LE	Z == 1 || N != V	знаковый	Op1 <= Op2

Сравнения

- CMP Rn, Op2
  - Res := Rn - Op2
- CMN Rn, Op2
  - Res := Rn + Op2
- TST Rn, Op2
  - Res := Rn & Op2
- TEQ Rn, Op2
  - Res := Rn ^ Op2

- Пример:

```
cmp     r0, #1
movlo   r0, #1   /* r0 := 1 if r0 <= 1 */
```

Пример

```
    .text
    .global  sgn
    .type    sgn, %function
sgn:
    cmp      r0, #0
    movlt    r0, #-1      /* mvnlt r0, #0 */
    movgt    r0, #1
    bx       lr
    .size    sgn, .-sgn
```

MOV

- MOV{COND}{S} Rd, Op2
  - Rd := Op2
- MVN{COND}{S} Rd, Op2
  - Rd := 0xFFFFFFFF xor Op2
- Примеры:

```
mov     r2, r3
mvn     r1, r0      /* r1 = ~r0 */
mov     r0, r0      /* do nothing */
movlt   r0, #-1     /* r0 := -1 if Res < 0 */
mvnlt   r0, #0      /* r0 := -1 if Res < 0 */
```

Immediates

- В ARM immeadiate может принимать значения вида Byte ROR Shift*2
  - Byte - 8 битное значение
  - Shift - 4 битное значение
  - ROR - циклический сдвиг вправо
- Допустимые значения:
  - 0x00000FF0
  - 0x000000FF
  - 0xF000000F
  - 0xFF000000
- Недопустимые значения:
  - 0x0F0F0000
  - 0x000001FF
  - 0x55550000

Арифметика

- ADD Rd, Rn, Op2
  - Rd := Rn + Op2
- ADC Rd, Rn, Op2
  - Rd := Rn + Op2 + C
- SUB Rd, Rn, Op2
  - Rd := Rn - Op2
- SBC Rd, Rn, Op2
  - Rd := Rn - Op2 - !C
- RSB Rd, Rn, Op2
  - Rd := Op2 - Rn
- RSC Rd, Rn, Op2
  - Rd := Op2 - Rn - !C
- Примеры:

```
add     r0, r1, r2      /* r0 := r1 + r2 */
sub     r5, r3, #10     /* r5 := r3 - 10 */
rsb     r2, r5, #0xff   /* r2 := 0xFF - r5 */
```

Арифметика

- MUL{COND}{S} Rd, Rm, Rs
  - Rd := Rm * Rs
- MLA{COND}{S} Rd, Rm, Rs, Rn
  - Rd := Rm * Rs + Rn
- Rm и Rs - НЕ ДОЛЖНЫ БЫТЬ ОДИНАКОВЫМИ!
- Пример:

```
        .text
        .global   sqr
        .type     sqr, %function
sqr:
        movs      r1,  r0
        mulne     r0,  r1,  r0
        bx        lr
        .size     sqr, .-sqr
```

Функции

- Аргументы функций:
  - регистры r0-r4
  - через стек
- Возвращаемое значение:
  - регистр r0
- Вызов функции:
  - B{COND} Addr
    - pc := Addr
  - BL{COND} Addr
    - pc := Addr, lr := адрес инструкции, следующей за BL
  - B и BL осуществляют “короткие” переходы, в пределах 2^25 байт
  - Дальние переходы осуществляются в ручную:
    - MOV lr, pc
    - LDR pc, =Label

Пример

```
     .text
     .type     bar, %function
bar:
     mov       r0,  #0xff
     bx        lr
     .size     bar, .-bar
     .global   foo
     .type     foo, %function
foo:
     mov       r1,  lr        /* save return address */
     mov       lr,  pc        /* load return address in lr */
     ldr       pc,  =foo      /* far call to foo */
     bx        r1             /* return */
     .size     foo, .-foo
```

Работа с памятью

- Загрузка значения из памяти:
  - LDR{COND}{B,SB, H, SH} Rd, Addr
    - Rd := значение по адресу Addr
    - B, SB, H, SH - указывают тип (байт или 2 байта, знаковый или беззнаковый)
- Сохранение значения в память:
  - STR{COND}{B, H} Rd, Addr
    - Значение по адресу Addr := Rd
    - B, H - указывают тип (байт или 2 байта)
- Примеры:

```
ldr     r0, [r1]
ldrb    r0, [r1]
ldr     r0, [r1,#4]      /* r0 := value at address r1 + 4 */
str     r0, [r1],   #4   /* value at address r1:= r0, r1 := r1 + 4 */
str     r0, [r1,#4]!     /* value at r1 + 4 := r0, r1 := r1 + 4 */
```

Работа со стеком

- STMDB sp!, {Rn1, Rn2, Rn3-Rnk}
  - Rni - регистры сохраняемые в стек
  - M - multiple, позволяет сохранить сразу несколько регистров
  - DB - decrement before, перед использованием sp уменьшается
  - ! - обновление значения sp
- LDMIA sp!, {Rn1, Rn2, Rn3-Rnk}
  - Rni - регистры извлекаемые из стека
  - IA - increment after, после исользования sp увеличивается
- Пример:

```
stmdb    sp!,    {lr}    /* function prolog */
ldr      r0,     =msg    /* load msg address */
bl       puts            /* call c puts */
ldmia    sp!,    {pc}
```

Рекурсивный факториал

```
    .text
    .global   factorial
    .type     factorial, %function
factorial:
    cmp       r0,  #1
    movlo     r0,  #1                 /* r0 := 1 if arg < 1 */
    bxls      lr                      /* return if arg <= 1 */
    stmdb     sp!, {r0-r1,lr}         /* save args and lr */
    sub       r0,  r0,  #1
    bl        factorial               /* r0 := factorial(arg - 1) */
    ldmia     sp!, {r1}               /* r1 := arg */
    mul       r0,  r0,  r1            /* r0 := r1 * r0 [factorial(arg - 1) * arg] */
    ldmia     sp!, {r1,pc}            /* return */
    .size     factorial, .-factorial
```

Итеративный факториал

```
        .text
        .global    factorial
        .type      factorial, %function
factorial:
        cmp        r0,   #1
        movlo      r0,   #1
        bxls       lr
        mov        r1,   #1
loop:
        mul        r1,   r1,   r0
        subs       r0,   r0,   #1
        bne        loop
        mov        r0,   r1
        bx         lr
        .size      factorial, .-factorial
```

Системный вызов

```
    .text
    .global  sysprint
    .type    sysprint, %function
sysprint:
    stmdb    sp!,     {r0-r2,r7,lr}
    mov      r2,      r1       /* size passed in r1 */
    mov      r1,      r0       /* ptr passed in r0 */
    mov      r0,      #1       /* stdout == 1 */
    mov      r7,      #4       /* write syscall number = 4 */
    swi      #0                /* int 0x80 */
    ldmia    sp!,     {r0-r2,r7,pc}
    .size    sysprint, .-sysprint
```