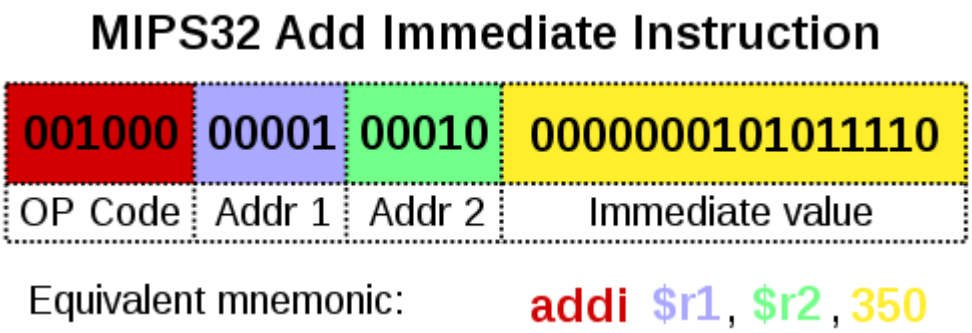
TUGAS ORGANISASI DAN ARSITEKTUR KOMPUTER
SET INSTRUCTION PROCESSOR INTEL AND ARM
NOVENDIUS EKA SAPUTRA [L200150084]

Daftar Intruksi Set Processor Intel

1. Bagian dari instruksi



Satu instruksi mungkin memiliki beberapa bidang, yang mengidentifikasi operasi logis untuk dilakukan, dan bisa juga menyertakan alamat sumber dan tujuan dan nilai-nilai konstan. Ini adalah MIPS "Tambahkan Segera" instruksi yang memungkinkan pemilihan register sumber dan tujuan dan inklusi dari sebuah konstanta kecil. pada arsitektur tradisional, instruksi mencakupopcode menentukan operasi yang akan dilakukan, seperti "isi menambah memori untuk mendaftar", dan nol atau lebih operanpenspesifikasi, yang dapat menentukanregister, memori lokasi, atau data harfiah. Para penspesifikasi operan mungkin memilikimode pengamatan menentukan makna mereka atau mungkin dalam bidang tetap. Dalam [kata sangat panjang instruksi [

(VLIW) arsitektur, yang mencakup banyak microcode arsitektur, opkode simultan dan operan yang ditentukan dalam sebuah instruksi.

Beberapa set instruksi eksotis tidak memiliki bidang opcode (seperti Transportasi Dipicu Arsitektur (TTA) atau mesin Forth maya), hanya operan (s). Lainnya tidak biasa " 0-operan" set instruksi kekurangan dalam suatu operan bidang specifier, seperti beberapa [mesin [tumpukan]] termasuk NOSC / balik / NOSC /.

2. AAA—ASCII Adjust After Addition

Penggunaan: AAA

Memodifikasi Flags: AF CF (OF, PF, SF, ZF terdefinisi)

Perubahan isi AL ke desimal dibongkar valid

Opcode	Instruction	Op/En	64-bit Mode	Compat/Leg Mode
37	AAA	NP	Invalid	Valid

Instruction Operand Encoding

Op/En	Operand 1	Operand 2	Operand 3	Operand 4
NP	NA	NA	NA	NA

Operasi

```
IF 64-Bit Mode
    THEN
         #UD;
    ELSE
         IF ((AL AND 0FH) > 9) or (AF = 1)
              THEN
                   AL ← AL + 6;
                   AH ← AH + 1;
                   AF ← 1;
                   CF ← 1;
                   AL ← AL AND 0FH;
              ELSE
                   AF ← 0;
                   CF ← 0; AL ← AL AND 0FH;
         FI;
FI;
```

3. AAD—ASCII Adjust AX Before Division

Penggunaan: AAD

Memodifikasi Flags: SF ZF PF (AF, CF, OF terdefinisi)

Digunakan sebelum membagi angka desimal dibongkar. Mengalikan AH dengan 10 dan menambahkan hasil ke AL. Set AH ke nol. Instruksi ini juga dikenal memiliki perilaku tak tercatat.

AL: = 10 * AH + AL
AH: = 0

Opcode	Instruction	Op/En	64-bit Mode	Compat/Leg Mode
D5 0A	AAD	NP	Invalid	Valid
D5 *ib*	AAD imm8	NP	Invalid	Valid

Instruction Operand Encoding

Op/En	Operand 1	Operand 2	Operand 3	Operand 4
NP	NA	NA	NA	NA

Operasi

```
IF 64-Bit Mode
    THEN
         #UD;
    ELSE
         tempAL ← AL;
         tempAH ← AH;
         AL ← (tempAL + (tempAH * imm8)) AND FFH; (* imm8 is set to
0AH for the AAD mnemonic.*)
         AH ← 0;
FI;
```

4. AAM—ASCII Adjust AX After Multiply

Penggunaan: AAM

Memodifikasi Flags: PF SF ZF (AF, CF, OF terdefinisi)

Digunakan setelah perkalian dua angka desimal dibongkar, instruksi ini menyesuaikan jumlah desimal dibongkar. Urutan menggigit tinggi setiap byte harus memusatkan perhatian sebelum menggunakan instruksi ini. Instruksi ini juga dikenal memiliki perilaku tak tercatat.

Opcode	Instruction	Op/En	64-bit Mode	Compat/Leg Mode
D4 0A	AAM	NP	Invalid	Valid
D4 *ib*	AAM imm8	NP	Invalid	Valid

Instruction Operand Encoding

Op/En	Operand 1	Operand 2	Operand 3	Operand 4
NP	NA	NA	NA	NA

Operasi

```
IF 64-Bit Mode
    THEN
         #UD;
    ELSE
         tempAL ← AL;
         AH ← tempAL / imm8; (* imm8 is set to 0AH for the AAM mnemonic *)
         AL ← tempAL MOD imm8;
FI;
```

5. **AAS—ASCII Adjust AL After Subtraction**

Penggunaan: AAS

Memodifikasi Flags: AF CF (OF, PF, SF, ZF terdefinisi)

Mengoreksi hasil dari pengurangan desimal dibongkar sebelumnya di AL.

Opcode	Instruction	Op/En	64-bit Mode	Compat/Leg Mode
3F	AAS	NP	Invalid	Valid

Instruction Operand Encoding

Op/En	Operand 1	Operand 2	Operand 3	Operand 4
NP	NA	NA	NA	NA

Operasi

```
IF 64-bit mode
    THEN
         #UD;
    ELSE
         IF ((AL AND 0FH) > 9) or (AF = 1)
              THEN
                   AX ← AX - 6;
                   AH ← AH - 1;
                   AF ← 1;
                   CF ← 1;
                   AL ← AL AND 0FH;
              ELSE
                   CF ← 0;
                   AF ← 0;
                   AL ← AL AND 0FH;
         FI;
FI;
```

6. ADC—Add with Carry

Penggunaan: ADC dest, src

Memodifikasi Flags: AF CF OF SF PF ZF

Merangkum dua operan biner menempatkan hasilnya di tujuan. Jika CF diatur, 1 ditambahkan ke tujuan.

Instruction Operand Encoding

Op/En	Operand 1	Operand 2	Operand 3	Operand 4
RM	ModRM:reg (r, w)	ModRM:r/m (r)	NA	NA
MR	ModRM:r/m (r, w)	ModRM:reg (r)	NA	NA
MI	ModRM:r/m (r, w)	imm8	NA	NA
I	AL/AX/EAX/RAX	imm8	NA	NA

	jam			
operan	286	386	486	**ukuran Bytes**
reg, reg	2	2	1	2
mem, reg	7	7	3	2-4
reg, mem	7	6	2	2-4
reg, immed	3	2	1	3-4
mem, immed	7	7	3	3-6
accum, immed	3	2	1	2-3

Operasi

```
DEST ← DEST + SRC + CF;
```

ADD - Arithmetic Addition

AND - Logical And

ARPL - Adjusted Requested Privilege Level of Selector (286+ PM)

BOUND - Array Index Bound Check (80188+)

BSF - Bit Scan Forward (386+)

BSR - Bit Scan Reverse (386+)

BSWAP - Byte Swap (486+)

BT - Bit Test (386+)

BTC - Bit Test with Compliment (386+

BTR - Bit Test with Reset (386+)

BTS - Bit Test and Set (386+)

CALL - Procedure Call

CBW - Convert Byte to Word

CDQ - Convert Double to Quad (386+)

CLC - Clear Carry

CLD - Clear Direction Flag

CLI - Clear Interrupt Flag (disable)

CLTS - Clear Task Switched Flag (286+ privileged).

CMC - Complement Carry Flag

CMP - Compare

CMPS - Compare String (Byte, Word or Doubleword)

CMPXCHG - Compare and Exchange

CWD - Convert Word to Doubleword

CWDE - Convert Word to Extended Doubleword (386+)

DAA - Decimal Adjust for Addition

DAS - Decimal Adjust for Subtraction

DEC - Decrement.

DIV - Divide

ENTER - Make Stack Frame (80188+)

ESC - Escape

HLT - Halt CPU

IDIV - Signed Integer Division

IMUL - Signed Multiply

IN - Input Byte or Word From Port

INC - Increment

INS - Input String from Port (80188+)

INT - Interrupt

INTO - Interrupt on Overflow

INVD - Invalidate Cache (486+)

INVLPG - Invalidate Translation Look-Aside Buffer Entry (486+)

IRET/IRETD - Interrupt Return.

Jxx - Jump Instructions Table.

JCXZ/JECXZ - Jump if Register (E)CX is Zero

JMP - Unconditional Jump

LAHF - Load Register AH From Flags

LAR - Load Access Rights (286+ protected)

LDS - Load Pointer Using DS

LEA - Load Effective Address

LEAVE - Restore Stack for Procedure Exit (80188+)

LES - Load Pointer Using ES

LFS - Load Pointer Using FS (386+)

LGDT - Load Global Descriptor Table (286+ privileged)

LIDT - Load Interrupt Descriptor Table (286+ privileged)

LGS - Load Pointer Using GS (386+)

LLDT - Load Local Descriptor Table (286+ privileged)

LMSW - Load Machine Status Word (286+ privileged)

LOCK - Lock Bus

LODS - Load String (Byte, Word or Double)

LOOP - Decrement CX and Loop if CX Not Zero

LOOPE/LOOPZ - Loop While Equal / Loop While Zero

LOOPNZ/LOOPNE - Loop While Not Zero / Loop While Not Equal

LSL - Load Segment Limit (286+ protected)

LSS - Load Pointer Using SS (386+)

LTR - Load Task Register (286+ privileged)

MOV - Move Byte or Word

MOVS - Move String (Byte or Word)

MOVSX - Move with Sign Extend (386+)

MOVZX - Move with Zero Extend (386+)

MUL - Unsigned Multiply

NEG - Two's Complement Negation

NOP - No Operation (90h)

NOT - One's Compliment Negation (Logical NOT)

OR - Inclusive Logical OR

OUT - Output Data to Port.

OUTS - Output String to Port (80188+)

POP - Pop Word off Stack

POPA/POPAD - Pop All Registers onto Stack (80188+)

POPF/POPFD - Pop Flags off Stack

PUSH - Push Word onto Stack

PUSHA/PUSHAD - Push All Registers onto Stack (80188+)

PUSHF/PUSHFD - Push Flags onto Stack

RCL - Rotate Through Carry Left.

RCR - Rotate Through Carry Right

REP - Repeat String Operation

REPE/REPZ - Repeat Equal / Repeat Zero

REPNE/REPNZ - Repeat Not Equal / Repeat Not Zero

RET/RETF - Return From Procedure

ROL - Rotate Left.

ROR - Rotate Right

SAHF - Store AH Register into FLAGS

SAL - Shift Arithmetic Left / Shift Logical Left

SAR - Shift Arithmetic Right

SBB - Subtract with Borrow/Carry

SCAS - Scan String (Byte, Word or Doubleword)

SETAE/SETNB - Set if Above or Equal / Set if Not Below (386+)

SETB/SETNAE - Set if Below / Set if Not Above or Equal (386+)

SETBE/SETNA - Set if Below or Equal / Set if Not Above (386+)

SETE/SETZ - Set if Equal / Set if Zero (386+)

SETNE/SETNZ - Set if Not Equal / Set if Not Zero (386+)

SETL/SETNGE - Set if Less / Set if Not Greater or Equal (386+)

SETGE/SETNL - Set if Greater or Equal / Set if Not Less (386+)

SETLE/SETNG - Set if Less or Equal / Set if Not greater or Equal (386+)

SETG/SETNLE - Set if Greater / Set if Not Less or Equal (386+)

SETS - Set if Signed (386+)

SETNS - Set if Not Signed (386+)

SETC - Set if Carry (386+)

SETNC - Set if Not Carry (386+)

SETO - Set if Overflow (386+)

SETNO - Set if Not Overflow (386+)

SETP/SETPE - Set if Parity / Set if Parity Even (386+)

SETNP/SETPO - Set if No Parity / Set if Parity Odd (386+)

SGDT - Store Global Descriptor Table (286+ privileged)

SIDT - Store Interrupt Descriptor Table (286+ privileged)

SHL - Shift Logical Left

SHR - Shift Logical Right

SHLD/SHRD - Double Precision Shift (386+)

SLDT - Store Local Descriptor Table (286+ privileged)

SMSW - Store Machine Status Word (286+ privileged)

STC - Set Carry

STD - Set Direction Flag

STI - Set Interrupt Flag (Enable Interrupts)

STOS - Store String (Byte, Word or Doubleword)

STR - Store Task Register (286+ privileged)

SUB - Subtract

TEST - Test For Bit Pattern

VERR - Verify Read (286+ protected)

VERW - Verify Write (286+ protected)

WAIT/FWAIT - Event Wait

WBINVD - Write-Back and Invalidate Cache (486+)

XCHG - Exchange

XLAT/XLATB - Translate

XOR - Exclusive OR

Intruksi Set Processor Arm

ARM

- Advanced RISC Machines (ARM) limited telah mendesain suatu famili mikroprosesor dan melisensikan desain tersebut ke perusahaan lain untuk fabrikasi chip yang penggunaannya dalam produk komputer dan sistem uang embedded.

- Perusahan ARM yang relatif baru, merupakan perkembangan dari perusahan Acorn Computer yang mengembangkan desain prosesor pada awal tahun 1980-an.

- Penggunaan utama mikroprosesor ARM adalah pada aplikasi embedded yang berdaya rendah dan berbiaya rendah, seperti misalnya mobile telephone, modem komunikasi, sistem manejemen mesin mobil, dan hand-held

1. Instruksi Akses Memory Dan Mode Pengalamatan

- Eksekusi Conditional Instruksi

Fitur yang membedakan dan agak tidak biasa dari prosesor ARM adalah semua instruksinya dieksekusi secara conditional, tergantung pada kondisi yang ditetapkan pada instruksi tersebut. Instruksi tersebut dieksekusi hanya jika keadaan saat ini dari conditional code flag prosesor memenuhi kondisi yang ditetapkan dalam bit b_{31-28} dari instruksi tersebut. Jika tidak prosesor melanjutkan ke instruksi berikutnya salah satu kondisi tersebut digunakan untuk mengindikasikan bahwa instruksi tersebut selalu dieksekusi

- Mode Pengalamatan Memori

Metode dasar untuk mengalamati operand memori adalah membangkitkan effective address, EA, dari operan tersebut dengan menambahkan offset bertanda keisi base register Rn, yang ditentukan dalam instruksi. Besarnya offset tersebut dapat berupa nilai immediate yang terdapat dalam 12 bit low order instruksi atau isi dari register ketiga, Rm, yang dinamai dengan 4 bit low order tanda arah offset terdapat dalam field OP-code.

- OPERAND LOAD/STORE MULTIPLE

Selain instruksi load dan store untuk operand tunggal, terdapat 2 instruksi untuk me-load dan menyimpan banyak operand. Instruksi itu disebut instruksi transfer block. Sub set apapun dari general purpose register load atau disimpan. Hanya operand word yang diperbolehkan, dan OP code yang digunakan dalam load multiple dan store multiple. Operand memori harus berada dalam lokasi word yang berurutan.

2. Instruksi Move Register

Meng-copy isi satu register ke register lain atau untuk me-load nilai immediate ke suatu register. Instruksi move

MOV Rd , Rm

3. Instruksi Aritmatika Dan Logika

Set instruksi ARM memiliki sejumlah instruksi untuk operand aritmatika dan logika pada operand yang berada dalam general-purpose register atau dinyatakan sebagai operand immediate dalam instruksi itu sendiri.

Terdapat instruksi untuk operand logika AND,OR,NOT,XOR, dan bit-clear. Instruksi seperti compare disediakan untuk men-set condition code flag berdasarkan hasil dari operasi aritmatika dan logika pada dua operand

A. Intruksi Aritmatika

Ekspresi bahasa assembly dasar untuk instruksi aritmatika adalah

Opcode Rd, Rn, Rm

Dimana operasi yang ditetapkan oleh OP code dilakukan menggunakan operand dalam general-purpose register Rn dan Rm. Hasilnya diletakkan dalam register Rd. Misalnya, instruksi

ADD R0, R2, R4

Menjalankan operasi

R0 ← [R2] + [R4]

Dan instruksi

SUB R0, R6, R5

Menjalankan operasi

R0 ← [R6] – [R5]

B. Intruksi Logika

Operasi logika AND, OR, XOR, dan Bit-clear diimplementasikan oleh instruksi OP code AND, ORR, EOR, dan BIC. Kode tersebut memiliki format yang sama dengan instruksi aritmatika. Instruksi

AND Rd, Rn, Rm

Menjalankan operasi

Rd ← [Rn] ^ [Rm]

Yang merupakan bitwise logical AND antara operand dalam register Rn dan Rm. Misalnya, jika register R0 berisi pola hexadesimal 02FA62CA dan R1 berisi pola 0000FFFF, maka instruksi

AND R0, RO, R1

Akan menyebabkan pola 000062CA diletakkan dalam register R0.

4. Intruksi Branch

Instruksi branch CONDITIONAL berisi offset 24-bit, 2'-complement, bertanda yang ditambahkan ke isi ter-update Program Counter untuk menghasilkan alamat target branch.

Instruksi Branch dieksekusi dengan cara yang sama seperti instruksi ARM yang lain, yaitu dieksekusi hanya jika keadaan terbaru condition code flag berhubungan dengan kondisi ditetapkan dalam field codition instruksi tersebut.

A. Setting Condition Code

beberapa instruksi, seperti compare, dinyatakan sebagai berikut

CMP Rn, Rm

Yang menjalankan operasi

[Rn] – [Rm]

Memiliki tujuan utama untuk men-set condition code flag berdasar pada hasil operasi pengurangan.

B. Program Loop untuk penambahan bilangan

Operasi load dan store dilakukan oleh instruksi pertama, kedua, dan terakhir yang digunakan oleh mode pengalamatan relative. Ini mengasumsikan bahwa lokasi memori N, pointer, dan SUM terdapat dalam rentang yang terjangkau oleh offset relatif terhadap PC. Lokasi memori pointer berisi alamat NUM 1 dari bilangan pertama yang akan ditambahkan, N berisi jumlah entri didalam list dan SUM digunakan untuk menyimpan jumlah tersebut

5. Bahasa Assembly

Bahasa assembly ARM memiliki assembler directive untuk menyiapkan ruang penyimpanan, menetapkan nilai numerik ke label alamat dari simbol konstanta, menentukan dimana program dan blok data akan ditempatkan dalam memori, menetapkan akhir teks source program fasilitas tersebut didekskripsikan secara umum.

6. Kontrol Aliran Program

- Condition Code Flag

68000 memiliki lima condition code flag, disimpan dalam register status. Selain flag M,Z,V, dan C yang dideskripsikan pada bagian 2.4.6, 6800 memiliki lima flag, X (extend). Di-set dengan cara yang sama dengan flag C, tetapi tidak dipengaruhi oleh banyak instruksi.

- Instruksi Branch

Instruksi conditional branch menyebabkan eksekusi program berlanjut dengan instruksi pada alamat target branch jika kondisi branch dipenuhi.

7. Register dan Pengalamatan

Dalam arsitektur 1A-32, memori adalah byte addressable menggunakan alamat 32-bit, dan instruksi beroperasi pada operand data 8 dan 32 bit. Ukuran operand ini disebut byte dan doubleword dalam istilah intel.

A. Struktur Register 1A-32
   - Terdapat delapan floating-point register untuk menyimpan operand data floating point doubleword atau quadword (64 bit). Floating-point register yang memiliki field ekstensi untuk menyediakan panjang total 80 bit.

   - Arsitektur 1A-32 berbasis pada model memori yang menghubungkan area yang berbeda di dalam memori, yang disebut segmen dengan kegunaan yang berbeda.

B. Mode Pengalamatan 1A-32
   - Arsitektur 1A-32 memiliki set mode pengalamatan yang besar dan fleksibel. Mode tersebut didesain untuk mengakses item data individu atau item data yang merupakan aggota dari list yang berurutan yang mulai pada alamat memori tertentu.
   - Mode dasar, yang tersedia pada kebanyakan prosesor telah dideskripsikan. Mode tersebut adalah : Immediate, Absolute, Register, dan Register indirect.