Lab ARM Introduction

José Nelson Amaral

Assignment

- MIPStoARM:
  - Arguments:
    - $a0 = pointer to memory containing a MIPS function that ends with the word 0x FFFF FFFF
  - Return Values:
    - $v0 = The number of ARM instructions generated
    - $v1 = A pointer to the first ARM instruction that is stored in memory space that you allocate.

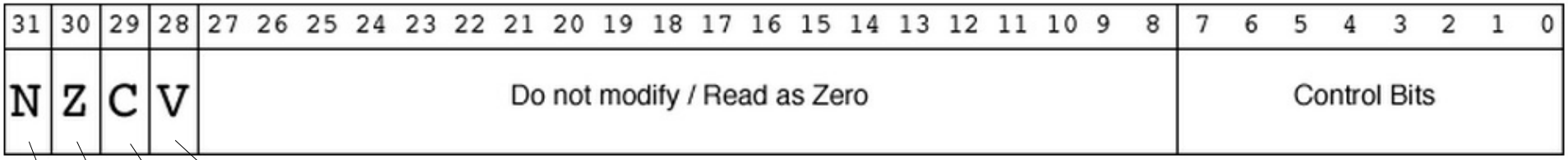
MIPS Instructions to Translate

Instruction	Encoding
ANDI $t, $s, imm	0011 00ss ssst tttt iiii iiii iiii iiii
AND $d, $s, $t	0000 00ss ssst tttt dddd d000 0010 0100
ORI $t, $s, imm	0011 01ss ssst tttt iiii iiii iiii iiii
OR $d, $s, $t	0000 00ss ssst tttt dddd d000 0010 0101
ADDI $t, $s, imm	0010 00ss ssst tttt iiii iiii iiii iiii
ADD $d, $s, $t	0000 00ss ssst tttt dddd d000 0010 0000
SUB $d, $s, $t	0000 00ss ssst tttt dddd d000 0010 0010
BEQ $s, $t, offset	0001 00ss ssst tttt iiii iiii iiii iiii
BGEZ $s, offset	0000 01ss sss0 0001 iiii iiii iiii iiii
SRA $d, $t, h	0000 0000 000t tttt dddd dhhh hh00 0011
SRL $d, $t, h	0000 0000 000t tttt dddd dhhh hh00 0010
SLL $d, $t, h	0000 0000 000t tttt dddd dhhh hh00 0000
SRLV $d, $t, $s	0000 00ss ssst tttt dddd d000 0000 0110
SLLV $d, $t, $s	0000 00ss ssst tttt dddd d000 0000 0100
JR $s	0000 00ss sss0 0000 0000 0000 0000 1000

The ARM Status Register

Instructions that have a **status bit** on change the condition codes in the status register

The CPSR (Current Program Status Register)



Set if signed operation resulted in **Overflow**

Set if operation produced a **Carry** bit

Set if operation result was **Zero**

Set if operation result was **Negative**

ARM Instruction Formats

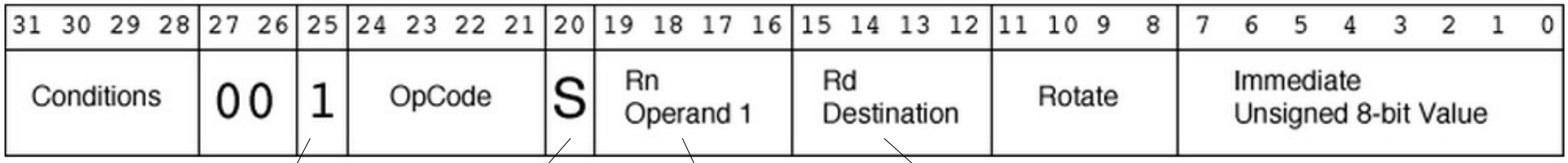
Data-Processing Immediate Format

31 30 29 28	27 26	25	24 23 22 21	20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0
Conditions	00	1	OpCode	S	Rn Operand 1	Rd Destination	Rotate	Immediate Unsigned 8-bit Value

Code	Flag Check	Meaning
1110	No check	Always Execute
0000	Z Set	Equals (As evaluated with CMP, below)
1010	N equals V	Greater Than or Equal to (As evaluated with CMP)
1100	Z clear AND (N equals V)	Greater Than (As evaluated with CMP)

ARM Instruction Formats

Data-Processing Immediate Format



Destination register.

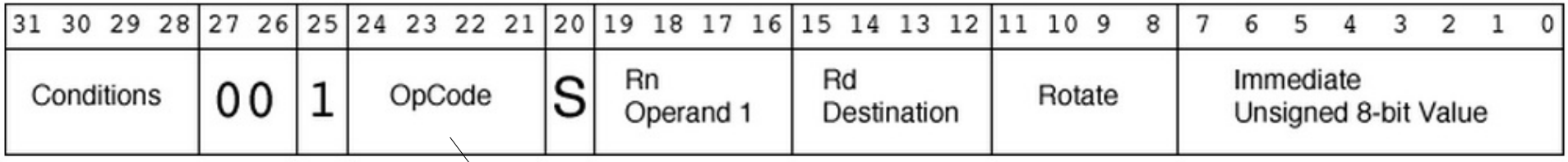
First operand register.

Status indicate if condition codes in CPSR should change.

Immediate
- 0: Operands are two registers
- 1: Operands are a register and immediate value

ARM Instruction Formats

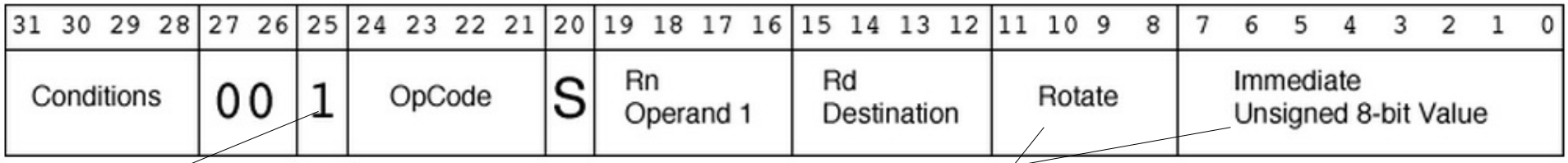
Data-Processing Immediate Format



Instruction	Opcode	Effect
AND	0000	operand1 AND operand2
OR	1100	operand1 OR operand2
ADD	0100	operand1 + operand2
SUB	0010	operand1 - operand2
MOV	1101	operand2
CMP	1010	operand1 - operand2, result not saved. (flags only)

ARM Instruction Formats

Data-Processing Immediate Format



31 30 29 28	27 26	25	24 23 22 21	20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0
Conditions	00	1	OpCode	S	Rn Operand 1	Rd Destination	Rotate	Immediate Unsigned 8-bit Value

The instruction has a 32-bit immediate value that is formed by:

Immediate 8-bit value

Rotated to the right by 2 times **Rotate** with wrap-around over 32 bits

Example:

8-bit immediate: 0110 1101

Rotate: $1110_2 = 14_{10}$

32-bit immediate: 0000 0000 0000 0000 0000 0110 1101 0000

ARM Instruction Formats

Data-Processing Register Format

31 30 29 28	27 26	25	24 23 22 21	20	19 18 17 16	15 14 13 12	11 10 9 8 7 6 5 4	3 2 1 0
Conditions	00	0	OpCode	S	Rn Operand 1	Rd Destination	Shift *	Rm Operand 2

The second operand is the content of **Rm** shifted as per **Shift**:

Rotate Right 0 shifts Rm one to the right and places Carry Flag from CPSR in bit 31.

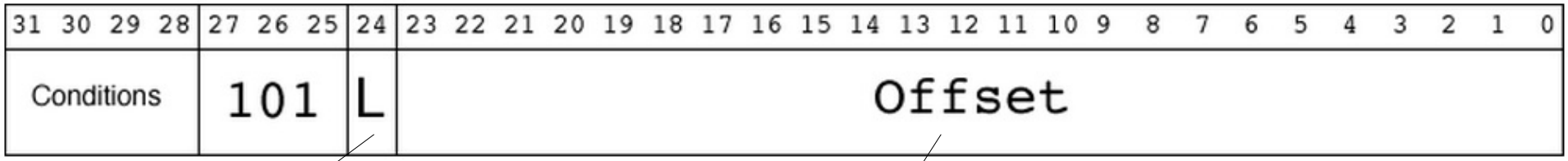
Shift (bits 11-4)

11 10 9 8 7	6 5	4
Shift Amount	Shift Type	0

Shift Types	
00	Logical Left
01	Logical Right
10	Arithmetic Right
11	Rotate Right

ARM Instruction Formats

Branch Format



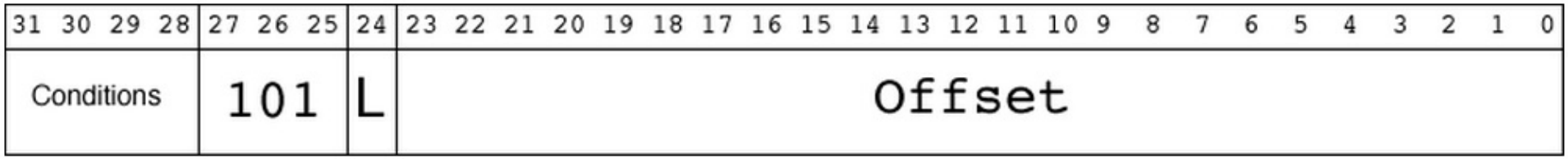
Indicates if address of next instruction should be stored into the link register (R14).

Shifted left by two and sign-extended.

ARM prefetches two instructions in advance, thus PC is set to the address of the branch instruction +8.

ARM Instruction Formats

Branch Format



A single conditional branch in MIPS may require two ARM instructions:

- One to set the condition
- Another for the branch

  → Changes in the offset of all branches

ARM Instruction Formats

Branch Exchange Format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Conditions	0 0 0 1	0 0 1 0	1 1 1 1	1 1 1 1	1 1 1 1	0 0 0 1	Rn Address Reg

Equivalent to the jr instruction in MIPS

Specifies the register for the relative jump

Register Translation

MIPS Register	ARM Register
Reserved for Translator	R0
$at ($1)	R1
$v0 ($2)	R2
$v1 ($3)	R3
$a0 ($4)	R4
$a1 ($5)	R5
$a2 ($6)	R6
$a3 ($7)	R7
$t0 ($8)	R8
$t1 ($9)	R9
$t2 ($10)	R10
$t3 ($11)	R11
$t4 ($12)	R12
$sp ($29)	R13
$ra ($31)	R14

Suggested Algorithm to Translate Branches

MIPS Binary (shown disassembled)

0x00100100	0x2042FFFF
0x00100104	0x20210000
0x00100108	0x0441FFF4
0x0010010c	0x0421FFFC

MIPS-to-ARM Table

0x00100200	
0x00100204	
0x00100208	
0x0010020c	

Branch Table

0x00100300	
0x00100304	
0x00100308	
0x0010030c	

ARM Instructions

0x00100400	
0x00100404	
0x00100408	
0x0010040c	
0x00100410	
0x00100414	

First pass is over the MIPS instruction binary codes

Incorrect Suggested Algorithm to Translate Branches

MIPS Binary (shown disassembled)

0x00100100	addi $2, $2, -1
0x00100104	addi $1, $1, 1
0x00100108	bgez $2, -12
0x0010010c	beq $1, $1, -4

MIPS-to-ARM Table

0x00100200	0x00100400
0x00100204	0x00100404
0x00100208	0x0010040c
0x0010020c	0x00100414

Branch Table

0x00100300	0
0x00100304	0
0x00100308	0x00100200
0x0010030c	0x0010020c

Target Instr. Address 0x00100400
Branch Instr. Address – 0x0010040c
– 0x0000000c
PC Offset – 0x00000008
-20

Target Instr. Address 0x00100414
Branch Instr. Address – 0x00100414
– 0x00000000
PC Offset – 0x00000008
-8

ARM Instructions

0x00100400	SUB R2, R2, 1
0x00100404	ADD R1, R1, 1
0x00100408	CMP S R2, 0
0x0010040c	BGE -20
0x00100410	CMP S R1, R1
0x00100414	BEQ -8

Problem with previous solution

- The offset generated for the last branch in the segment is incorrect because the branch target is the branch instruction itself (a weird program that creates an infinite loop).
- A simple solution is to store in the MIPS-to-ARM table the address of the ARM branch (instead of the address of the ARM compare instruction as in the previous solution).
- Now the address calculation has to compensate for the existence of the CMP instruction by adding 4 to the address found in the MIPS-to-ARM table before subtracting from the target instruction address.

Limitation

- One limitation of this new solution (shown in the next slide) is that it assumes that every MIPS branch will be translated into two ARM instructions.
- If that is not the case, then another single-bit table can be used to indicate for each branch if the branch got translated to a single or to two instruction and then do the correct computation of the offset accordingly.

Correction to Suggested Algorithm to Translate Branches (assumes that every MIPS branch is translated to two ARM instructions)

MIPS Binary (shown disassembled)

0x00100100	addi $2, $2, -1
0x00100104	addi $1, $1, 1
0x00100108	bgez $2, -12
0x0010010c	beq $1, $1, -4

MIPS-to-ARM Table

0x00100200	0x00100400
0x00100204	0x00100404
0x00100208	0x00100408
0x0010020c	0x00100410

Branch Table

0x00100300	0
0x00100304	0
0x00100308	0x00100200
0x0010030c	0x0010020c

Target Instr. Address 0x00100400

Branch Instr. Address – 0x0010040c # 0x00100408 + 4

– 0x0000000c

PC Offset – 0x00000008

-20

Target Instr. Address 0x00100410

Branch Instr. Address – 0x00100414 # 0x00100410 + 4

– 0x00000004

PC Offset – 0x00000008

-12

ARM Instructions

0x00100400	SUB R2, R2, 1
0x00100404	ADD R1, R1, 1
0x00100408	CMP S R2, 0
0x0010040c	BGE -20
0x00100410	CMP S R1, R1
0x00100414	BEQ -12