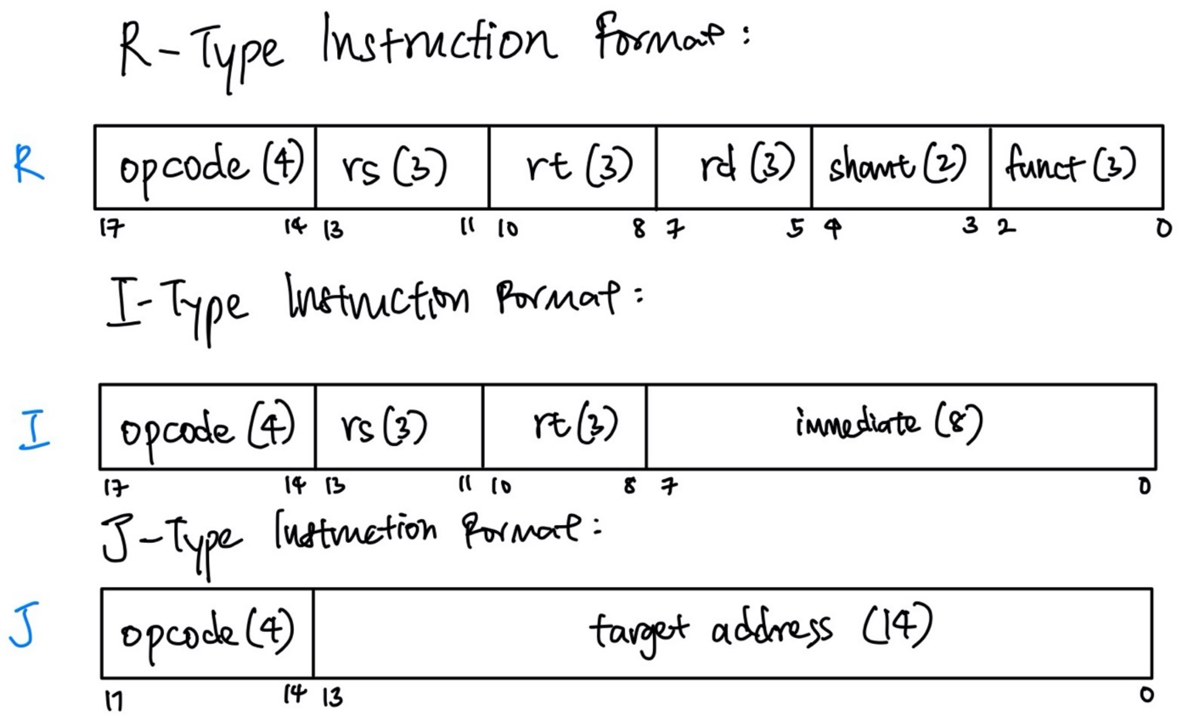
TIC2401 Term Paper

Ansel Lim, [redacted]

The ANSEL processor is inspired by the MIPS processor. However, unlike the MIPS processor, it uses 18 bit-wide instructions. Apart from this design quirk, the ANSEL processor is remarkably similar to the MIPS processor.

The ANSEL processor has eight different registers, specified by \$0, \$1, …, \$7. The \$0 register actually contains the constant zero. The other 7 registers may be used for anything else, e.g. values for function results and expression evaluation or temporaries (similar to, say, \$v0, \$t0, \$s0 in MIPS).

Data words in the ANSEL processor are 32 bits wide, just like in MIPS! As in MIPS, the ANSEL processor supports R-type, I-type, and J-type instructions, as shown here.


R-Type Instruction Format:
R
opcode (4) | rs (3) | rt (3) | rd (3) | shamt (2) | funct (3)
17 14 13 11 10 8 7 5 4 3 2 0
I-Type Instruction Format:
I
opcode (4) | rs (3) | rt (3) | immediate (8)
17 14 13 11 10 8 7 0
J-Type Instruction Format:
J
opcode (4) | target address (14)
17 14 13 0

The numbers in the parentheses indicate the number of bits allocated. For example, opcode takes up four bits.

The ANSEL processor has a small instruction set supporting basic arithmetic and logical operations.

INSTRUCTION SET OF THE ANSEL PROCESSOR

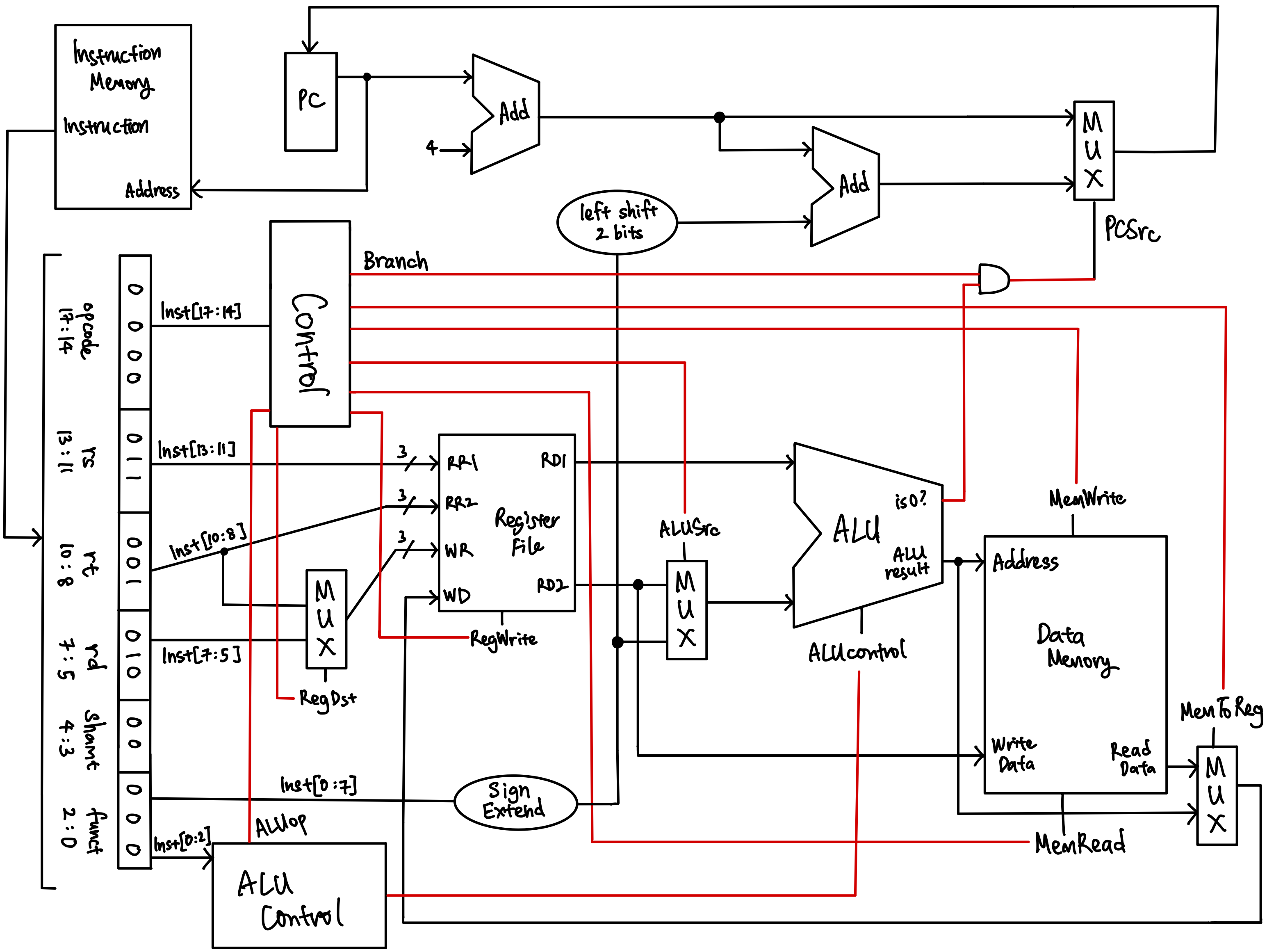
Name	Mnemonic	Type	Operation	Opcode (decimal)	Funct (decimal)

Add	add	R	R[rd]=R[rs]+R[rt]	0	0
Subtract	sub	R	R[rd]=R[rs]+R[rt]	1	1
And	and	R	R[rd]=R[rs]&R[rt]	2	2
Or	or	R	R[rd]=R[rs]|R[rt]	3	3
Set Less Than	slt	R	R[rd] = (R[rs]<R[rt]) ? 1:0	4	4
Shift Left Logical	sll	R	R[rd]=R[rt]<<shamt	5	5
Shift Right Logical	srl	R	R[rd]=R[rt]>>shamt	6	6
Add Immediate	addi	I	R[rt]=R[rs]+SignExtImm	7	Not applicable
And Immediate	andi	I	R[rt]=R[rs]&ZeroExtImm	8	Not applicable
Or Immediate	ori	I	R[rt]=R[rs]|ZeroExtImm	9	Not applicable
Set Less Than Imm.	slti	I	R[rt]=(R[rs]<SignExtImm) ? 1:0	10	Not applicable
Branch On Equal	beq	I	if(R[rs]==R[rt]) PC=PC+4+BranchAddr	11	Not applicable
Branch On Not Equal	bne	I	if(R[rs]!=R[rt]) PC=PC+4+BranchAddr	12	Not applicable
Load Word	lw	I	R[rt]=M[R[rs]+SignExtImm]	13	Not applicable
Store Word	sw	I	M[R[rs]+SignExtImm]=R[rt]	14	Not applicable
Jump	j	J	PC=JumpAddr	15	Not applicable

In MIPS, the opcode for R-type instructions is 0. This is not the case in the ANSEL processor. I decided to just have different opcodes since the 4-bit width of opcode allows more than enough unique values.

Note that the ANSEL processor DOES NOT HAVE:

- integer multiplication: including the multiply operation would make finding the square of a number a trivial problem; in lieu of a multiply operation, a user can simply add the multiplicand to itself (b-1) times, where b is the multiplier, and the user will arrive at the same result.
- integer division…although this could be implemented by discarding one of the less useful operations, say for example, slti, and replacing this operation with the a \`divide\` operation \`div\`. This would be an R-format operation R[rd]=R[rs]/R[rt] that only keeps the integer quotient.


Instruction Memory
Instruction
Address
PC
Add
4
Add
MUX
left shift 2 bits
PCSrc
Control
Branch
Inst[17:14]
opcode 17:14
rs 13:11
rt 10:8
rd 7:5
shamt 4:3
funct 2:0
0 0 0 0
0 1 1
0 0 1
0 1 0
0 0
0 0 0
Inst[13:11]
Inst[10:8]
Inst[7:5]
3
3
3
RR1
RR2
WR
WD
RD1
RD2
Register File
RegWrite
MUX
RegDst
ALUSrc
MUX
ALU
is0?
ALU result
ALUcontrol
MemWrite
Address
Data Memory
Write Data
Read Data
MemToReg
MUX
MemRead
Inst[0:7]
Sign Extend
Inst[0:2]
ALUop
ALU Control

- the special HI and LO registers that MIPS has for storing the results of multiplication or division.
- support for floating-point arithmetic.
- the `nor` or `xor` logical operations: these may be implemented using the other logical operators.

Take a number N and return N^2. Let's suppose that N is a word variable located at the memory address specified in $7.

Instruction	Line number	Explanation
lw $1, 0($7)	#1	Loads N into register $1
add $2, $0, $0	#2	Create a 'result variable' $2
add $3, $0, $0	#3	Initializes counter variable in $3
loop: beq $3, $1, Exit	#4	Start loop, check for break condition
add $2, $2, $1	#5	Add N to the result variable
addi $3, $3, 1	#6	Increment counter variable
beq $0, $0, loop	#7	Go back to start of loop
Exit:	#8	-
sw $2, 0($7)	#9	Store result into memory

Equivalent C code:

Equivalent C code:

```
int main(void){
    int N = 100;
    int temp = 0;
    int i = 0;
    while (i != N){
      temp += N;
      i++;
    }
    N = temp;
    return 1;
}
```

C variable	Register
N	$1
temp	$2
i	$3

Line Number	Instruction type	Machine Code in Binary
#(1) lw \$1, 0(\$7) — opcode =13, rt, immediate, rs	I	$\underbrace{1101}_{opcode=13}\ \underbrace{111}_{rs=7}\ \underbrace{001}_{rt=1}\ \underbrace{00000000}_{immediate=0}$ Load the value of N from memory.
#(2) add \$2, \$0, \$0 — opcode =0, rd, rs, rt	R	$\underbrace{0000}_{opcode=0}\ \underbrace{000}_{rs=0}\ \underbrace{000}_{rt=0}\ \underbrace{010}_{rd=2}\ \underbrace{00}_{shamt=0}\ \underbrace{000}_{funct=0}$ Initialize result register
#(3) add \$3, \$0, \$0 — rd, rs, rt	R	$\underbrace{0000}_{opcode=0}\ \underbrace{000}_{rs=0}\ \underbrace{000}_{rt=0}\ \underbrace{011}_{rd=3}\ \underbrace{00}_{shamt=0}\ \underbrace{000}_{funct=0}$ initialize counter
#(4) loop: beq \$3, \$1, Exit — opcode =11, rs, rt, BranchAddr	I	$\underbrace{1011}_{opcode=11}\ \underbrace{010}_{rs=3}\ \underbrace{001}_{rs=1}\ \underbrace{00000011}_{BranchAddr=3}$ since Exit is at line 8, and this instruction is at line 4.
#(5) add \$2, \$2, \$1 — rd, rs, rt	R	$\underbrace{0000}_{opcode=0}\ \underbrace{010}_{rs}\ \underbrace{001}_{rt}\ \underbrace{010}_{rd}\ \underbrace{00}_{shamt}\ \underbrace{000}_{funct}$ Add N to the result register
#6 addi \$3, \$3, 1 — opcode =7, rt, rs, immediate	I	$\underbrace{0111}_{opcode=7}\ \underbrace{011}_{rs=3}\ \underbrace{011}_{rs=3}\ \underbrace{00000001}_{immediate=1}$ Increment the counter.
#7 beq \$0, \$0, loop — opcode =11, rs, rt, BranchAddr	I	$\underbrace{1011}_{opcode=11}\ \underbrace{000}_{rs=0}\ \underbrace{000}_{rt=0}\ \underbrace{11111100}_{immediate=-4}$ since PC+4 points to line #8, and loop is at line #4. Loop back.
#9 sw \$2, 0(\$7) — opcode =14, rt, immediate, rs	I	$\underbrace{1110}_{opcode=14}\ \underbrace{111}_{rs=7}\ \underbrace{010}_{rt=2}\ \underbrace{00000000}_{immediate=0}$ Store answer at same memory location.