

KGP-RISC DOCUMENTATION

Group 8

SUHAS JAIN - 19CS30048

MONAL PRASAD - 19CS30030

Instruction Set Architecture:

| Class | Instruction | Usage | Meaning |
|------------|---------------------------------|----------------|---|
| Arithmetic | Add | add rs,rt | $rs \leftarrow (rs) + (rt)$ |
| | Comp | comp rs,rt | $rs \leftarrow 2's \text{ Complement } (rs)$ |
| | Add immediate | addi rs,imm | $rs \leftarrow (rs) + imm$ |
| | Complement Immediate | compi rs,imm | $rs \leftarrow 2's \text{ Complement } (imm)$ |
| Logic | AND | and rs,rt | $rs \leftarrow (rs) \wedge (rt)$ |
| | XOR | xor rs,rt | $rs \leftarrow (rs) \oplus (rt)$ |
| Shift | Shift left logical | shll rs, sh | $rs \leftarrow (rs)$ left-shifted by sh |
| | Shift right logical | shrl rs, sh | $rs \leftarrow (rs)$ right-shifted by sh |
| | Shift left logical variable | shllv rs, rt | $rs \leftarrow (rs)$ left-shifted by (rt) |
| | Shift right logical | shrlv rs, rt | $rs \leftarrow (rs)$ right-shifted by (rt) |
| | Shift right arithmetic | shra rs, sh | $rs \leftarrow (rs)$ arithmetic right-shifted by sh |
| | Shift right arithmetic variable | shrav rs, rt | $rs \leftarrow (rs)$ arithmetic right-shifted by (rt) |
| Memory | Load Word | lw rt,imm(rs) | $rt \leftarrow mem[(rs) + imm]$ |
| | Store Word | sw rt,imm,(rs) | $mem[(rs) + imm] \leftarrow (rt)$ |
| Branch | Unconditional branch | b L | goto L |
| | Branch Register | br rs | goto (rs) |
| | Branch on less than 0 | bltz rs,L | if(rs) < 0 then goto L |
| | Branch on flag zero | bz rs,L | if (rs) = 0 then goto L |
| | Branch on flag not zero | bnz rs,L | if(rs) ≠ 0 then goto L |
| | Branch and link | bl L | goto L; 31 ← (PC)+4 |
| | Branch on Carry | bcy L | goto L if Carry = 1 |
| | Branch on No Carry | bncy L | goto L if Carry = 0 |

Instruction Description:

The register usage convention for the available 32 registers is,

| Register number | Register name | Description |
|-----------------|---------------|-------------------------|
| 0 | \$zero | Zero value |
| 1-2 | \$v0 - \$v1 | Function return values |
| 3-7 | \$a0 - \$a4 | Function parameters |
| 8-21 | \$t0 - \$t13 | Temporary registers |
| 22-29 | \$s0 - \$s7 | Callee saved registers |
| 30 | \$sp | Stack pointer |
| 31 | \$ra | Function return address |

Instruction Set Architecture Format:

R type instructions

Format:

| OP code | Destination register | Source Register | Shift Amount | Function code |
|---------|----------------------|-----------------|--------------|---------------|
| 6 bits | 5 bits | 5 bits | 5 bits | 11 bits |

Functions:

| Instruction | OP code (6) | R1 (5) | R2 (5) | Shift amount (5) | Function Code (11) |
|--|-------------|--------|--------|------------------|--------------------|
| add | 000000 | rs | rt | XXXXX | 00000000000 |
| comp | 000000 | rs | rt | XXXXX | 00000000001 |
| and | 000000 | rs | rt | XXXXX | 00000000010 |
| xor | 000000 | rs | rt | XXXXX | 00000000011 |
| Shift left logical | 000000 | rs | xxxxx | Shift amount sh | 00000000100 |
| Shift right logical | 000000 | rs | xxxxx | Shift amount sh | 00000000101 |
| Shift left logical variable | 000000 | rs | rt | XXXXX | 00000000110 |
| Shift right logical variable | 000000 | rs | rt | XXXXX | 00000000111 |
| Shift right arithmetic | 000000 | rs | xxxxx | Shift amount sh | 00000001000 |
| Shift right arithmetic variable | 000000 | rs | rt | XXXXX | 00000001001 |

Number of Instructions supported:

Since we have a 11 bit function code 2^{11} instructions can be supported. Out of them we have used 10 instructions. So $2^{11} - 10 = 2038$ more instructions can be added later.

I type instructions

Format:

| OP code | Destination register | Source Register | Immediate Value |
|---------|----------------------|-----------------|-----------------|
| 6 bits | 5 bits | 5 bits | 16 bits |

Functions:

| Instruction | OP code (6) | R1 (5) | R2 (5) | Immediate Value (16) |
|----------------|----------------|-----------|-----------|--------------------------|
| Load word | 000001 | rs | rt | Immediate constant value |
| Store word | 000010 | rs | rt | Immediate constant value |
| Add immediate | 000011 | rs | XXXXX | Immediate constant value |
| Comp immediate | 000100 | rs | XXXXX | Immediate constant value |

Number of Instructions supported:

Since we don't have support for a function code instructions can be added only by varying op code. We have a 6 bit opcode out of which we are using 4 for I type instructions. So $2^6 - 4 = 60$ more instructions remain. However, branch type instructions are also derived from these 60 remaining opcode values.

Branch type instructions

Format:

| OP code | Destination register | Offset |
|---------|----------------------|---------|
| 6 bits | 5 bits | 21 bits |

Functions:

| Instruction | OP code (6) | R1 (5) | Offset (21) |
|--------------------------|----------------|-----------|----------------|
| Branch register | 000101 | rs | offset |
| Branch on less than zero | 000110 | rs | offset |
| Branch on flag zero | 000111 | rs | offset |
| Branch on flag not zero | 001000 | rs | offset |
| Unconditional Branch | 001001 | XXXXX | offset |

| | | | |
|---------------------------|--------|-------|--------|
| Branch and link | 001010 | XXXXX | offset |
| Branch on carry | 001011 | XXXXX | offset |
| Branch on no carry | 001100 | XXXXX | offset |

Number of Instructions supported:

Very similar to the I type instructions. After I type instructions 60 different opcodes remained unused out of which 8 have been used here. So $60 - 8 = 52$ more instructions can be added for later use.

Truth Table

R type instructions (opcode : 000000)

| Instruction | Function code | alu_control | ab_set | regWrite | MemWrite | MemRead | const_src | ALU_src | reg_data | reg_to_pc | regWrite_select |
|---------------------|---------------|-------------|--------|----------|----------|---------|-----------|---------|----------|-----------|-----------------|
| add rs,rt | 0000 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| comp rs,rt | 0001 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| and rs,rt | 0010 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| xor rs,rt | 0011 | 3 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| shll rs, sh | 0100 | 4 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| shrl rs, sh | 0101 | 5 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| shllv rs,rt | 0110 | 4 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| shrlv rs, rt | 0111 | 5 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| shra rs,sh | 1000 | 6 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| shrav rs,rt | 1001 | 6 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

I type instructions

| Instruction | opcode | alu_control | ab_set | regWrite | MemWrite | Mem Read | const_src | ALU_src | reg_data | reg_to_pc | regWrite_select |
|---------------------|--------|-------------|--------|----------|----------|----------|-----------|---------|----------|-----------|-----------------|
| addi rs, imm | 000001 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| compi rs,imm | 000010 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

| | | | | | | | | | | | |
|--------------------------|--------|---|---|---|---|---|---|---|---|---|---|
| lw rt,imm(rs) | 000011 | 9 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| sw rt,imm(rs) | 000100 | 9 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

Branch type instructions

| Instruction | opcode | alu_control | ab_set | regWrite | MemWrite | Mem Read | const_src | ALU_src | reg_data | reg_to_pc | regWrite_select |
|-------------------|--------|-------------|--------|----------|----------|----------|-----------|---------|----------|-----------|-----------------|
| br rs | 000101 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| bltz rs, L | 000110 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| bz rs, L | 000111 | 7 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| bnz rs, L | 001000 | 7 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| b L | 001001 | 8 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| bl L | 001010 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| bcy L | 001011 | 8 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| bncy L | 001100 | 8 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

Architecture Diagram and Datapath:

The PDF containing all the architecture of the whole processor is attached below, for simplicity of understanding and better detail, a top module is shown and each of the individual blocks of the top module are drawn in detail afterwards.

Also note that the control lines are drawn in orange in the diagrams whereas all the other lines are wires are drawn in black.









