uPOWER ISA MANUAL

ISA Version 0.1, January 2020

For use in CS250, NITK, Surathkal

uPOWER ISA is a (very small) subset of the POWER ISA v3.0. It is a 64bit ISA . All registers are 64 bits (numbered 0 (MSB) to 63 (LSB)).

The basic classes of instructions are as follows: Arithmetic and Logical Instructions, Data transfer Instructions, Compare and Branch instructions, Shift/Rotate Instructions. uPOWER ISA contains only integer instructions. Most ALU instructions operate on byte, half-word, word, and doubleword, operands from the general purpose registers. The uPower ISA uses instructions that are four bytes long and word-aligned. A set of 32 General Purpose Registers (GPRs), each of 64bits are available (Fig. 2). Signed integers are represented in two's complement form. There are no computational instructions that modify elements in memory. To use an operand from memory in a computation and then modify the same or another memory location, the contents of the storage operand must be loaded into a register, modified, and then stored back to the target location. Hence, uPower may be called a Load-Store ISA. Fig. 1 is a logical representation of instruction processing.

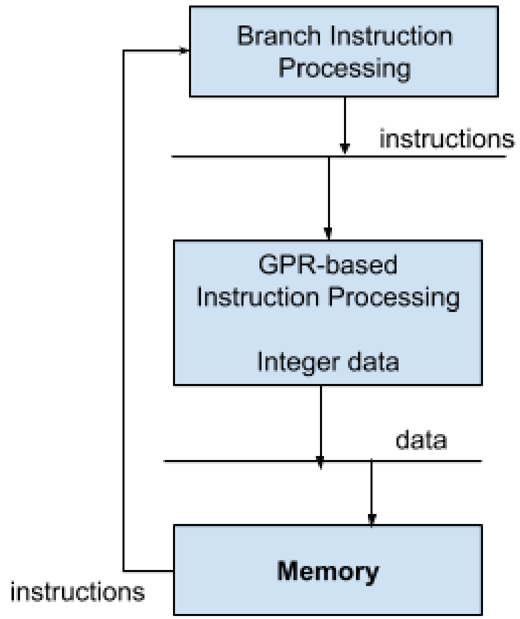


Fig. 1. Logical processing model for the uPower ISA.

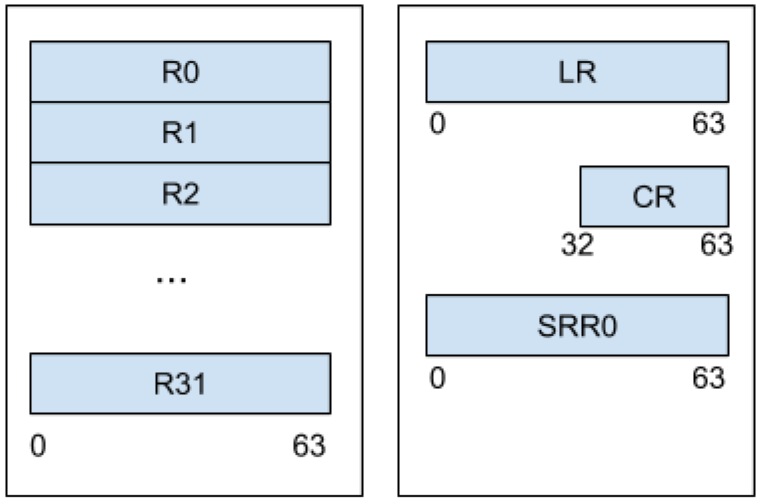


Fig. 2. General purpose register file, and other registers.

uPower ISA includes the following special registers (Fig. 2):

1. Link Register (LR) - 64 bits. Holds the return address after the function call instruction (bl).
2. Condition Register (CR) - 32 bits. CR is updated by compare instructions. CR tracks the result of the compare for later use by the conditional branch instructions. CR has eight 4-bit fields : CR0 (bits 32 … 35) … CR7 (bits 60 … 63). Use CR7 in uPOWER.
   Example usage of the CR. Consider the following code segment.
   ```
   cmpwi 7, 9, 4      # compare (9) and (4). update CR7 (check cmp, cmpi)
   bne 7, Loop        # if CR7 indicates non-equal, branch to label "Loop" (check instructions cmp, cmpi)
   ```
3. Special Register R0 (SRR0): stores the NIA after the System call instruction (sc).

REGISTER NAME, NUMBER, USE, CALL CONVENTION

R0, R1, …, R31 are General Purpose Registers. RA, RB, RS, RT - refer to one of the GPR.
R0 - Register 0. Implicit output operand for cmp family. R0 is used to store the System Call number.
R2 - Global Offset Table Pointer
R3 onwards - Inputs to System call
CIA: Current Instruction Address, which is the 64-bit address of the instruction being described by a sequence of RTL. Used by relative branches to set the Next Instruction Address (NIA), and by Branch instructions with LK=1 to set the Link Register. Does not correspond to any architected register. The CIA is sometimes referred to as the Program Counter (PC).
NIA: Next Instruction Address, which is the 64-bit address of the next instruction to be executed.
SRR0: Special Register R0. Contains the return address after system call is serviced. Used for nothing else now.

INSTRUCTION FORMATS

All instructions are 32 bits large. Bits are numbers 0 to 31. The instruction formats are shown below.

Format	Binary Encoding	Format	Binary Encoding
X	PO 0 | RS 6 | RA 11 | RB 16 | XO 21 | Rc 31	XO	PO 0 | RT 6 | RA 11 | RB 16 | OE 21 | XO 22 | Rc 31
XS	PO 0 | RS 6 | RA 11 | sh 16 | XO 21 | sh 30 | Rc 31		
D	PO 0 | RT 6 | RA 11 | SI 16	DS	PO 0 | RT 6 | RA 11 | DS 16 | XO 30
M	PO 0 | RS 6 | RA 11 | SH 16 | MB 21 | ME 26 | Rc 31		
B	PO 0 | BO 6 | BI 11 | BD 16 | AA 30 | LK 31	I	PO 0 | LI 6 | AA 30 | LK 31

Instruction Fields

Field	Description	Field	Description
PO	Primary Opcode Formats: All	AA	Absolute Address flag. Indicates if the address in the BD/LI fields is a relative filed (AA=0) or an absolute address (AA=1). Formats: B, I
RS/RT/RA/RB	RA: GPR as a source or as a target. Formats: D, DS, M, X, XO, XS RB: GPR to be used as a source. Formats: M, X, XO RS: GPR to be used as a source. Formats: RT: GPR to be used as a target.	SH	Shift Amount. Formats: M, X
XO	Extended Opcode	sh	Fields that are concatenated to specify a shift amount. Formats: XS

Rc	Record bit.		
MB	Field used in M-form instructions to specify the first 1-bit of a 64-bit mask, as described in Section 3.3.14, “Fixed-Point Rotate and Shift Instructions” on page 100. Formats: M	ME	Field used in M-form instructions to specify the last 1-bit of a 64-bit mask, as described in Section 3.3.14, “Fixed-Point Rotate and Shift Instructions” on page 100. Formats: M
SI	Signed integer immediate field.		
BO	Field used to specify options for the Branch Conditional instructions. Formats: B	BI	Field used to specify a bit in the CR to be tested by a Branch Conditional instruction. Formats: B
BD	Branch Displacement. Signed 2’s complement branch displacement concatenated on the right with 0b00 and sign-extended to 64 bits. Formats: B		

uPOWER INSTRUCTION SET

	Table: All instructions of the uPOWER ISA					
	Instruction	**Mnemonic & Usage**	**Meaning**	**Comment**	**Instruction Format**	**Binary Instruction Fields OP/OE/Rc/XO/AA/LK**
	Arithmetic Instructions					
1	add	add RT,RA,RB	RT = (RA) + (RB)	The sum (RA|0) + (SI || 0x0000) is placed into register	XO	31/0/0/266/-/-
2	add immediate	addi RT,RA,SI	if RA = 0 then RT = EXTS(SI) else RT = (RA) + EXTS(SI)	The sum (RA|0) + SI is placed into register RT.	D	14/-/-/-/-/-
3	Add Immediate Shifted	addis RT, RA, SI	if RA = 0 then RT = EXTS(SI) else RT = (RA) + EXTS(SI || $^{16}0$)	The sum (RA|0) + (SI || 0x0000) is placed into register RT	D	15/-/-/-/-
4	And	and RA.RS,RB	RA = RS and RB	The contents of register RS are ANDed with the contents of register RB and the result is placed into register RA.	X	31/--/--/28/--/--
5	And immediate	andi RA,RS,UI	RA←(RS)&($^{48}0$ || UI)	The contents of register RS are ANDed with $^{48}0$ || UI and the result is placed into register RA.	D	28/--/--/--/--/--
6	Extend Sign Word	extsw RA, RS	s = $(RS)_{32}$ $RA_{32:63}$ = $RS_{32:63}$ $RA_{0:31}$ = ^{32}s	$(RS)_{32:63}$ are placed into $RA_{32:63}$. $RA_{0:31}$ are filled with a copy of $(RS)_{32}$.	X	31/0/986/-/-
7	Nand	nand RA,RS,RB	RA = ¬((RS) & (RB))	The contents of register RS are ANDed with the contents of register RB and the complemented result is placed into register RA. nand or nor with RS=RB can be used to obtain the one's complement.	X	31/0/-/476/-/0

8	OR	or RA,RS,RB	RA←(RS) | (RB)	The contents of register RS are ORed with the contents of register RB and the result is placed into register RA. Extends: mr	X	31/0/-/444/-/0
9	Or Immediate	ori RA,RS,UI	RA = (RS) | (0 || UI)	The contents of register RS are ORed with 48 0 || UI and the result is placed into register RA. The preferred "no-op" (an instruction that does nothing) Is: ori 0,0,0	D	24/-/-/-/-
10	Subtract from	subf RT,RA,RB	RT = ¬ (RA) + (RB) + 1	RT ← RB - RA	XO	31/0/0/40/-/-
11	Exclusive or	xor RA RS RB	RA = RS xor RB	The contents of register RS are XORed with the contents of register RB and the result is placed into register RA.	X	31/0/-/316/-/--
12	Exclusive or immediate	xori RA,RS,UI	RA=(RS) XOR (^{48}0||UI)	Contents of register RS are XORed with ^{48}0||UI and the result is placed into register RA. NOP is xori 0,0,0	D	26/-/-/-/-/-
	Data transfer					
13	Load doubleword	ld RT,DS(RA)	if RA = 0 then b = 0 else b = (RA) EA = b + EXTS(DS || 0b00) RT = MEM(EA, 8)	EA is the sum (RA|0) + (DS||0b00). The doubleword in storage addressed by EA is loaded into RT.	DS	58/-/-/0/-/-
14	Load word & zero	lwz RT,D(RA)	if RA = 0 then b = 0 else b = (RA) EA = b + EXTS(D) RT = ^{32}0 || MEM(EA, 4)	EA is sum (RA|0)+ D. The word in storage addressed by EA is loaded into $RT_{32:63}$. $RT_{0:31}$ are set to 0.	D	32/--/--/--/--/--
15	Store doubleword	std RS,DS(RA)	if RA = 0 then b = 0 else b = (RA) EA = b + EXTS(DS || 0b00)	EA is the sum (RA|0)+ (DS||0b00). (RS) is stored into the doubleword in storage addressed by EA.	DS	62/--/--/0/--/--

			MEM(EA, 8) = (RS)			
16	Store word	stw RS,D(RA)	if RA = 0 then b = 0 else b = (RA) EA = b + EXTS(D) MEM(EA, 4) = $(RS)_{32:63}$	EA is the sum (RA|0)+ D. $(RS)_{32:63}$ are stored into the word in storage addressed by EA.	D	36/--/--/--/--/--
17	Store Word with Update	stwu RS, D(RA)	EA = (RA) + EXTS(D) MEM(EA, 4) = $(RS)_{32:63}$ RA = EA	The EA is the sum (RA) + D. $(RS)_{32:63}$ are stored into the word in storage addressed by EA. EA is placed into register RA. If RA=0, the instruction form is invalid.	D	37/--/--/--/--/--
18	Load halfword	lhz RT,D(RA)	if RA = 0 then b = 0 else b = (RA) EA = b + EXTS(D) RT = 0 || MEM(EA, 2)	EA is the sum (RA|0) + D. The halfword in storage addressed by EA is loaded into $RT_{48:63}$ (Least significant half word). $RT_{0:47}$ are set to 0.	D	40/--/--/--/--/--
19	Load Halfword Algebraic	lha RT,D(RA)	if RA = 0 then b ← 0 else b ← (RA) b + EXTS(D) EA ← EXTS(MEM(EA, 2))	EA is the sum (RA|0) + D. The halfword in storage addressed by EA is loaded into $RT_{48:63}$ (Least significant half word). $RT_{0:47}$ are filled with a copy of bit 0 of the loaded halfword.	D	42/--/--/--/--/--
20	Store halfword	sth RS,D(RA)	if RA = 0 then b = 0 else b = (RA) EA = b + EXTS(D) MEM(EA,2)= $(RS)_{48:63}$	EA is the sum (RA|0)+ D. $(RS)_{48:63}$ are stored into the halfword in storage addressed by EA.	D	44/--/--/--/--/--
21	Load byte and zero	lbz RT,D(RA)	if RA = 0 then b = 0 else b = (RA) EA = b + EXTS(D) RT = 0 || MEM(EA, 1)	EA is the sum (RA|0) + D. The byte in storage addressed by EA is loaded into $RT_{56:63}$. $RT_{0:55}$ are set to 0.	D	34/--/--/--/--/--
22	Store byte	stb RS,D(RA)	if RA = 0 then b = 0 else b = (RA) EA = b + EXTS(D) MEM(EA, 1) = (RS) 56:63	EA is the sum (RA|0)+ D. $(RS)_{56:63}$ are stored into the byte in storage addressed by EA.	D	38/--/--/--/--/--
	Shift/ Rotate					

23	Rotate Left Word Immediate then AND with Mask	rlwinm RA,RS,SH,MB,ME	$n \leftarrow SH$ $r \leftarrow ROTL_{32}((RS)_{32:63}, n)$ $m \leftarrow MASK(MB+32, ME+32)$ $RA \leftarrow r \& m$	The contents of register RS are rotated 32 left SH bits. A mask is generated having 1-bits from bit MB+32 through bit ME+32 and 0-bits elsewhere. The rotated data are ANDed with the generated mask and the result is placed into register RA.	M	21/-/0/-/-/-
25	Shift Left Doubleword	sld RA, RS, RB	$n \leftarrow (RB)_{58:63}$ $r \leftarrow ROTL_{64}((RS), n)$ if $(RB)_{57} = 0$ then $m \leftarrow MASK(0, 63-n)$ else $m \leftarrow {}^{64}0$ $RA \leftarrow r \& m$	Contents of register RS are shifted left the number of bits specified by $(RB)_{57:63}$ (least significant 7 bits). Bits shifted out of position 0 are lost. Zeros are supplied to the vacated positions on the right. The result is placed into register RA. Shift amounts from 64 to 127 give a zero result.	X	31/-/0/27/-/-
26	Shift Right Doubleword	srd RA, RS, RB	$n \leftarrow (RB)_{58:63}$ $r \leftarrow ROTL_{64}((RS),64-n)$ if $(RB)_{57} = 0$ then $m \leftarrow MASK(n, 63)$ else $m \leftarrow {}^{64}0$ $RA \leftarrow r \& m$	Contents of register RS are shifted right the number of bits specified by $(RB)_{57:63}$. Bits shifted out of position 63 are lost. Zeros are supplied to the vacated positions on the left. The result is placed into register RA. Shift amounts from 64 to 127 give a zero result.	X	31/-/0/539/-/-
27	Shift Right Algebraic Doubleword	srad RA, RS, RB	$n \leftarrow (RB)_{58:63}$ $r \leftarrow ROTL_{64}((RS),64-n)$ if$(RB)_{57} = 0$ then $m \leftarrow MASK(n, 63)$ else $m \leftarrow {}^{64}0$ $s \leftarrow (RS)_0$ $RA \leftarrow r\&m \| ({}^{64}s) \& \neg m$	Contents of register RS are shifted right. Shift amount = low-order seven bits of GPR RB. Bits shifted out of position 63 are lost. Bit 0 of RS is replicated to fill the vacated positions on the left. The result is placed into RA. Shift amounts from 64 to 127 give a result of 64 sign bits in GRP RA.	X	31/-/0/794/-/-
	Shift right arithmetic immediate	sradi RA, RS, SH	$n \leftarrow sh_5 \| \| sh_{0:4}$ $r \leftarrow ROTL_{64}((RS),64-n)$ $m \leftarrow MASK(n, 63)$ $s \leftarrow (RS)_0$ $RA \leftarrow r\&m \| ({}^{64}s)\&\neg m$	RS are shifted right SH bits. Bits shifted out of position 63 are lost. Bit 0 of RS is replicated to fill the vacated positions on the left. The result is placed into RA.	XS	31/-/0/413/-/-
28	**Control Statements**					

29	Unconditional Branch	b LI (AA=0) (Relative Addressing) ba LI (AA=1) (Absolute Addressing)	if AA then NIA = EXTS(LI || 0b00) else NIA = CIA + EXTS(LI || 0b00) if LK then LR = CIA + 4	NIA is calculated using the 24b LI field. If AA=1, then NIA is absolute If AA=0, then NIA is relative If AA=0, then NIA ← EXTS(LI || 0b00) + CIA High-order 32 bits of the branch target address set to 0 in 32-bit mode. If AA=1, then NIA ← EXTS(LI || 0b00) High-order 32 bits of the branch target address set to 0 in 32-bit mode.	I	18/-/-/-/AA/0
30	Function Call	bl LI	AA=0, LK=1 NIA = CIA + EXTS(LI || 0b00) LR = CIA + 4	Unconditional Jump and link. Stores CIA in the Link Register. High-order 32 bits of the branch target address set to 0 in 32-bit mode.	I	18/-/-/-/0/1
31	Function Return (See bclr pseudoinstruction)	bclr	BH=00 BO=1z1zz LK=0 NIA $\leftarrow_{iea}$ $LR_{0:61}$ || 0b00	BH=00 - Instruction is subroutine return BO=1z1zz (z bit is ignored) - Branch always Branch Conditional to Link Register. $LR_{0:61}$ || 0b00. Uses return address contained in the Link Register. The instruction is a subroutine return.	XL	19/-/-/-/-/0
32	Branch conditional Relative	bc BO,BI,target_addr	NIA $\leftarrow_{iea}$ CIA + EXTS(BD || 0b00)	BO is ignored in uPOWER. Use only CR7 (bits 60 … 63) for uPOWER. BI+32 specifies the Condition Register bit to be tested. BI = 28 to for greater than. Check if bit 60 is ON. BI = 29 to for less than. Check if bit 61 is ON. BI = 30 to for equal to. Check if bit 62 is ON. target_addr specifies the branch target address	B	19/-/-/-/0/0

33	Branch conditional Absolute	bca BO,BI,target_addr	NIA $\leftarrow_{iea}$ EXTS(BD || 0b00)	BO is ignored in uPOWER. Use only CR7 (bits 60 … 63) for uPOWER. BI+32 specifies the Condition Register bit to be tested. BI = 28 to for greater than. Check if bit 60 is ON. BI = 29 to for less than. Check if bit 61 is ON. BI = 30 to for equal to. Check if bit 62 is ON. target_addr specifies the branch target address	B	19/-/-/-/1/0
	Compare Instructions					
34	Compare	cmp BF,L,RA,RB	a = (RA); b = (RB) if a < b then c = 0b1000 else if a>b then c = 0b0100 else c = 0b0010 $CR_{60:63}$ = c	Use BF = 7. Use CR7 in uPOWER. The contents of register RA are compared with the contents of register RB treating the operands as signed integers. The result of the comparison is placed into CR field BF.	X	31/-/-/0/-/- L=1
35	Compare Immediate	cmpi BF,L,RA,SI	a $\leftarrow$ (RA) If a < EXTS(SI) then c $\leftarrow$ 0b1000 else if a > EXTS(SI) then c $\leftarrow$ 0b0100 else c $\leftarrow$ 0b0010 $CR_{60:63} \leftarrow$ c	Use BF = 7. Use CR7 in uPOWER. (RA) is compared with the sign-extended value of the SI field, treating the operands as signed integers. The result of the comparison is placed into CR field BF. Extended: cmpdi, cmpwi	D	11/-/-/-/-/- L=1
36	System Call	sc LEV	SRR0 $\leftarrow_{iea}$ CIA+4 NIA $\leftarrow$ 0x0000_0000_0000_0C00	The effective address of the instruction following the System Call instruction is placed into SRR0. The next instruction to be fetched from effective address 0x0000_0000_0000_0C00.	SC	17/-/-/-/-/-

Notations from the table are explained below:

(RA) = Contents of register RA
$(RA)_{32:63}$ = Lower word of register RA
EA: Effective address

EXTS(x): Result of extending x on the left with sign bits
MEM(x, y): Contents of a sequence of y bytes of storage. The sequence starts with the byte at address x+y-1 and ends with the byte at address x (Little-Endian byte ordering).

MEM(EA, 8): Access 8 bytes starting from EA
¬(RA): one's complement of the contents of register RA.
$^{64}0$: 64 zero bits (zero replicated 64 times).
| - or.
|| - concatenate. $^{48}0$ || UI means bits of UI are concatenated with 48 zero bits.
A period (.) as the last character of an instruction mnemonic means that the instruction records status information in certain fields of the Condition Register as a side effect of execution.
XER_{SO} : System Register XER, bit number SO.
$\leftarrow_{iea}$: Assignment of an instruction effective address. In 32-bit mode the high-order 32 bits of the 64-bit target address are set to 0.

ROTL $_{64}$(x, y): Result of rotating the 64-bit value x left y positions
MASK(x, y): Mask having 1s in positions x through y (wrapping if x > y) and 0s elsewhere
CIA: Current Instruction Address, which is the 64-bit address of the instruction being described by a sequence of RTL. Used by relative branches to set the Next Instruction Address (NIA), and by Branch instructions with LK=1 to set the Link Register. Does not correspond to any architected register. The CIA is sometimes referred to as the Program Counter (PC).
NIA: Next Instruction Address, which is the 64-bit address of the next instruction to be executed. For a successful branch, the next instruction address is the branch target address: in RTL, this is indicated by assigning a value to NIA.

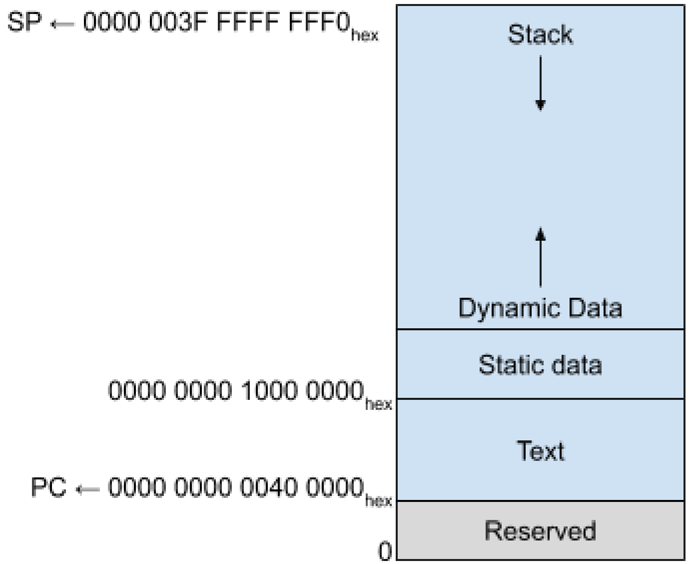
EXTENDED MNEMONICS (PSEUDOINSTRUCTIONS)

Extended Mnemonic	Pseudo Instruction	Translation	Comment
bclr	bclr 0, 0, 0	BH=0b00 means subroutine return	
beq	bc 12, 10, label	Branch on equal	Check CR7
blr	bclr LR	Branch unconditionally - bclr to LR	
bne	bc 4, 2, target	Branch on not equal	Check CR7
Compare Doubleword and Set bits (cmpd)	cmpd Rx, Ry	cmp 0,1,Rx,Ry	Compare and set bits
Compare Doubleword Immediate and Set bits (cmpdi)	cmpdi Rx,value	cmpi 0,1,Rx,value	
Load Immediate (li)	li 9, 10	addi 9, 0, 10 addit Rx, 0, Value	0 as second register operand means ignore
Load Immediate Shifted (lis)	lis 9, 10	addis 9, 0, 10 addis Rx, 0, value	Place 10 in upper 16 bits of the word
Load Address (la)	la Rx, D(Ry) la Rx, v	addi Rx,Ry,D addi Rx,Rv,Dv	
Move Register (mr)	mr 3, 6	or Rx, Ry, Ry	Move contents of Reg 6 into Reg 3

mtlr	mtlr Rx	mtspr 8, Rx	
mflr	mflr Rx	mfspr Rx,8	
Shift left Immediate	sldi Rx, Ry, n sldi 9, 9, 2	rldicr Rx,Ry,n,63-n rldicr 9, 9, 2, 61	
Shift left immediate	slwi Ra,Rs,n	rlwinm ra,rs,n,0,31-n	Shift the contents of register Rx left 8 bits, clearing the high-order 32 bits. slwi Rx,Rx,8
Subtract Immediate	subi Rx,Ry,value subis Rx,Ry,value	addi Rx, Ry, -value addis Rx,Ry,-value	
Subtract	sub Rx,Ry,Rz	subf Rx, Rz, Ry	Subtraction: Rx ← Ry - Rz
Executed NOP	xnop	xori 0, 0, 0	

MEMORY LAYOUT

Each program can access 2^{64} bytes of “effective address” (EA) space, subject to limitations imposed by the operating system. In a typical Power ISA system, each program's EA space is a subset of a larger “virtual address” (VA) space managed by the operating system. The stack looks like this:



ADDRESSING MODES

Addressing Mode	Instruction Examples	Comment
Immediate	li 3, 6 addi 2, 3, 25 ori 3, 6, 0b00000000000001	Input operand is a part of the instruction
Register	add 3, 4, 5	All input and output operands are registers
Direct		Instruction contains the address from which to load data. Used for global variable access, branching, and subroutine calls.
Relative		Calculates address based on PC. In short-range branches
Indexed		To access array elements for global variables
		It has two parts: a memory address and an *index register*. The index register is added to the specified address, and the result is used as the address for the memory access.
Base-Pointer	lbz 3, 0(2) lha 4, 1(31) std 5, 32(23) stb 6, 8(4)	Register has the base address and the literal number has the offset.