Bit32 Manual

Contents

Bit32 CPU Layout

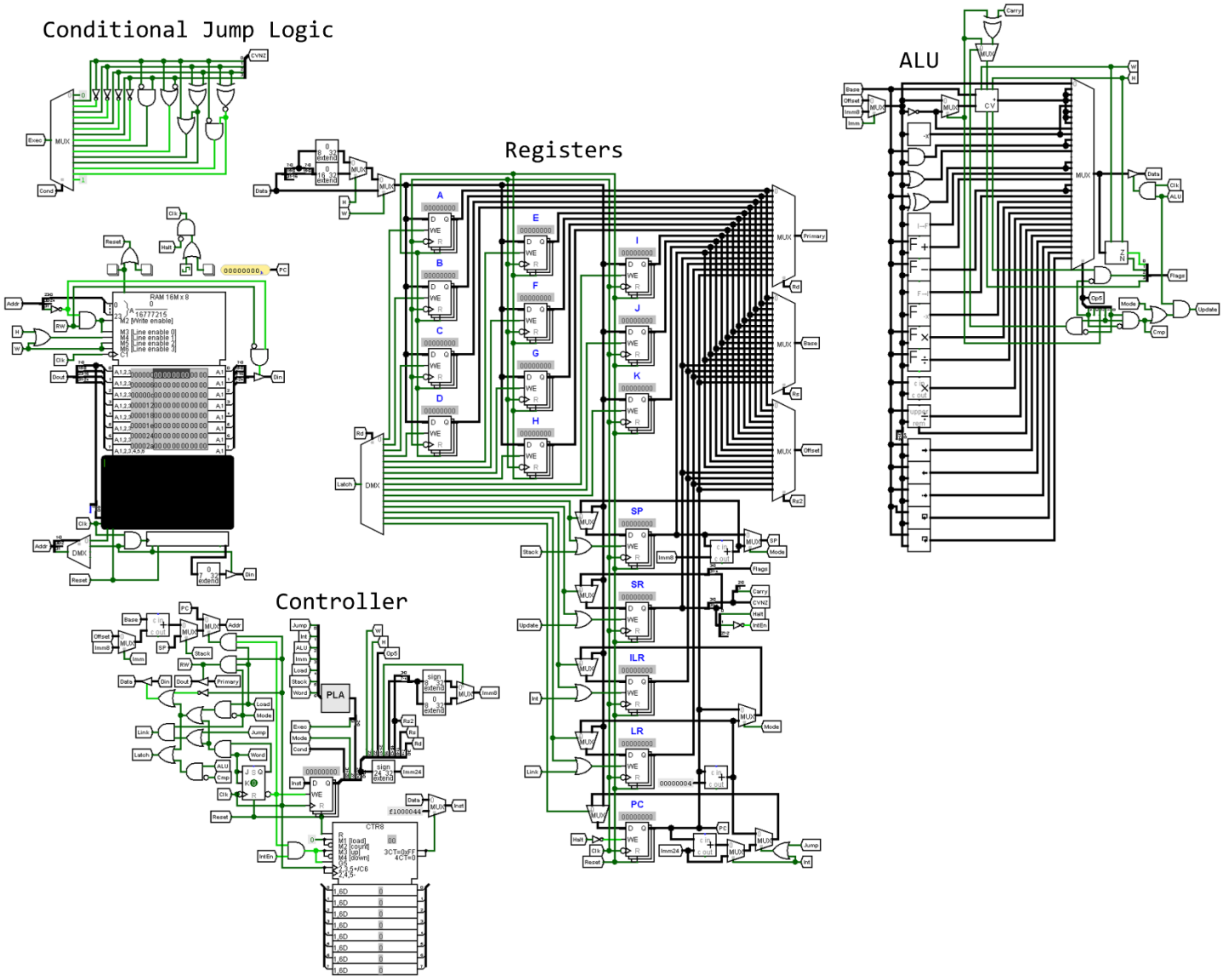


Figure 1: Layout of the bit32 CPU

Bit32 Specifications

Instructions and Cycle Times

Each instruction is 32 bits long. More information on the formats of each type of 32-bit instruction can be found later in this document. Every microcode instruction has a clock cycle of 1 except for the *load-intermediate* instruction which takes 2 clock cycles.

Registers

The bit32 CPU contains 16 addressable 32-bit registers: 11 general-purpose registers and 5 special purpose registers. The general-purpose registers can be used for anything including arithmetic and address calculations. There is also a 32-bit instruction register that stores the executing instruction

The stack pointer (SP) register is used to keep track of the top of the stack. The stack starts at the last address in RAM and grows downward.

The status register (SR) holds the status of the current program. This register dictates the halting state of the CPU as well as the flags for conditional instruction execution (see page 7).

The link register (LR) is used for calling and returning from procedures. LR is 32 bits and should only be used to calculate addresses for the RAM module. The interrupt link register (ILR) behaves the same as the LR but is only used for interrupt procedures specifically and should never be used with regular procedures.

The program counter (PC) register keeps track of the current instruction being executed. The program counter starts at 0 and is incremented after every instruction unless the instruction is a jump instruction in which case, the value in the PC becomes the target for the jump.

Code	Mnemonic	Purpose	Bits	Addressable
0	A	General Purpose	32	✔
1	B	General Purpose	32	✔
2	C	General Purpose	32	✔
3	D	General Purpose	32	✔
4	E	General Purpose	32	✔
5	F	General Purpose	32	✔
6	G	General Purpose	32	✔
7	H	General Purpose	32	✔
8	I	General Purpose	32	✔
9	J	General Purpose	32	✔
10	K	General Purpose	32	✔
11	SP	Stack Pointer	32	✔
12	SR	Status Register	32	✔
13	ILR	Interrupt Link	32	✔
14	LR	Link Register	32	✔
15	PC	Program Counter	32	✔
	IR	Instruction Register	32	

Table 1: bit32 CPU registers

Format Summary

The bit32 instruction set formats are shown in the following figure.

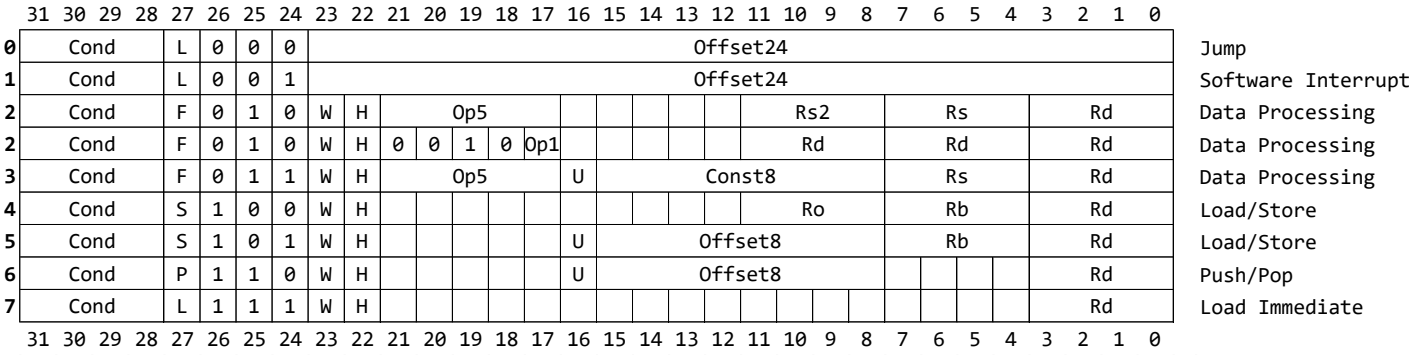


Figure 2: bit32 instruction set formats

There are 3 instruction encoding formats shown in the figure below.

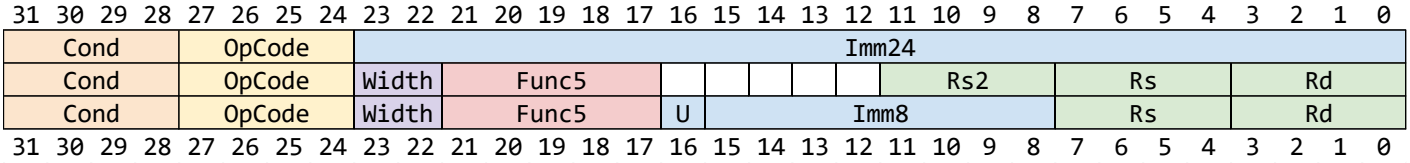


Figure 3: bit32 instruction encoding formats

Microcode Summary

The following section summarizes the bit32 instruction set microcode.

ALU Opcode Summary

The following table summarizes the bit32 instruction set operations.

Code	Mnemonic	Instruction	Condition Codes Set
0	MOV	Move to register	
1	ADD	Add	
2	ADC	Add with carry	
3	CMN	Compare negative	✔
4	MVN	Move Negative	
5	SUB	Subtract	
6	SBC	Subtract with carry	
7	CMP	Compare	✔
8	NOT	Bitwise not	
9	NEG	Arithmetic negate	
10	AND	Bitwise and	
11	TST	Test bits	✔
12	OR	Bitwise or	
13			
14	XOR	Bitwise exclusive or	
15	TEQ	Test equal	✔
16	ITF	Integer to float	
17	ADDF	Add float	
18	SUBF	Subtract float	
19	CMPF	Compare float	✔
20	FTI	Float to integer	
21	NEGF	Arithmetic negate float	
22	MULF	Multiply float	
23	DIVF	Divide float	
24	MUL	Multiply	
25	DIV	Divide	
26	MOD	Modulo	
27	SHR	Shift right	
28	SHL	Shift left	
29	ASL	Arithmetic shift left	
30	ROR	Rotate right	
31	ROR	Rotate left	

Table 2: bit32 instruction set opcodes

Load Operation Summary

The following table summarizes the Load operation for the bit32 instruction set.

Mnemonic	Instruction
LD	Load
ST	Store
LDI	Load Immediate

Table 3: bit32 load instruction

Conditional Instructions Summary

Each bit32 instruction can run conditionally depending on the state of the status register (SR). A conditional instruction can be represented by adding the suffix to the instruction mnemonic.

The following table summarizes the conditional execution codes for the bit32 instruction set.

Code	Mnemonic	Instruction
0	NV	Execute never
1	EQ	Execute if equal
2	NE	Execute if not equal
3	MI	Execute if minus
4	PL	Execute if plus
5	VS	Execute if overflow set
6	VC	Execute if overflow clear
7	CS/HS	Execute if carry set/higher or same
8	CC/LO	Execute if carry clear/lower
9	HI	Execute if higher
10	LS	Execute if lower or same
11	LE	Execute if less than or equal to
12	LT	Execute if less than
13	GT	Execute if greater than
14	GE	Execute if greater than or equal to
15	AL	Execute always

Table 4: bit32 conditional codes

Macrocode Summary

The following section summarizes the bit32 instruction set macrocode. Macrocode instructions are instructions that are not supported by the hardware and are made up of one or more microcode instructions.

Mnemonic	Instruction	Microcode

PUSH	Push register(s) to stack	`PUSH A, B =`
		`  PUSH A`
		`  PUSH B`

POP	Pop register(s) from stack	`POP A, B =`
		`  POP  B`
		`  POP  A`

CALL	Call procedure	`CALL A =`
		`  ADD  LR, PC, 8`
		`  MOV  PC, A`

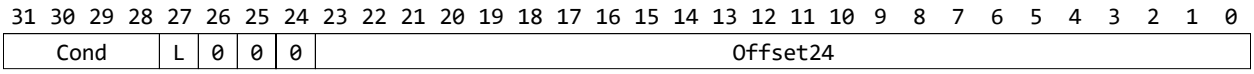
RET	Return from procedure	`RET =` `  MOV  PC, LR`
HALT	Halt execution	`HALT =` `  OR  SR, 0x20`

Table 5: bit32 macro code instructions

Micro Instructions Formats

The following section covers the 8 different instruction formats of the bit32 CPU instruction set.

Format 0: Jump



Cond: Condition code

L: Link bit (1 = CALL, 0 = JUMP)

Offset24: Immediate signed 24-bit offset value

Figure 4: Format 0

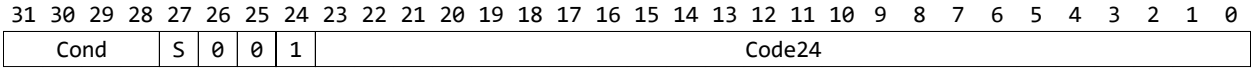
Operation

Instructions of this format perform a conditional relative jump. This means that the given immediate value is added to the program counter depending on the state of the flag register.

Examples

```
JMP   loop  ; Unconditionally jumps to the label “loop”
JNE   .L0   ; Jumps to the label “.L0” if the Z bit is clear
JGT   end   ; Jumps to label “end” if greater than
```

Format 1: Interrupt



Cond: Condition code

S: Software interrupt bit

Code24: Immediate 24-bit code

Figure 5: Format 1

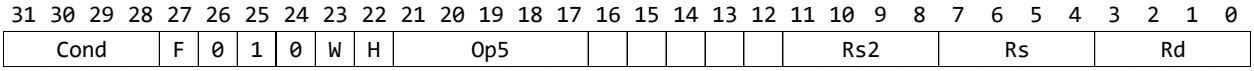
Operation

Instructions of this format perform a conditional absolute jump. This means that the given immediate value is moved to the program counter depending on the state of the flag register.

Examples

```
SWI   0x34  ; Software interrupt to code 0x34
```

Format 2: Ternary and Binary ALU Operation



Cond: Condition code

F: Flag bit

W: Word bit

H: Half-word bit

Op5: 5-bit opcode

Rd: Destination register

Rs: Source register

Rs2: Source register 2

Figure 6: Format 2

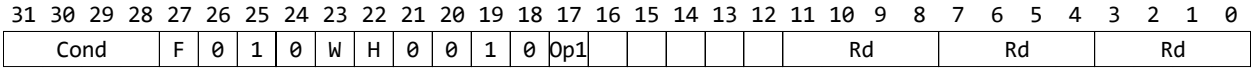
Operation

Instructions of this format perform a ternary or binary ALU operation on the given source registers and places the result in the destination register unless it's a comparison operation.

Examples

```
ADD   A, B, C     ; A = B + C
CMP   A, B  ; Subtract B from A and set the condition codes
OR    A, B  ; A = A | B
AND   A, B, C     ; A = B & C
MOV   A, B  ; Move the value in B to A
```

Format 2: Unary ALU Operation



Cond: Condition code

F: Flag bit

W: Word bit

H: Half-word bit

Op1: 1-bit opcode. (0 = NOT, 1 = NEG)

Rd: Source /destination register

Figure 7: Format 2 (Unary)

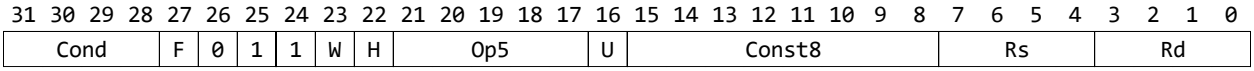
Operation

Instructions of this format perform a unary ALU operation on the given source register and place the result in the destination register.

Examples

```
NOT    A       ; A = ~A
NEG    A       ; A = -A
```

Format 3: Ternary and Binary Operation with Immediate



Cond: Condition code

F: Flag bit

W: Word bit

H: Half-word bit

Op5: 5-bit opcode

U: Unsigned bit (1 = unsigned, 0 = signed)

Const8: Immediate 8-bit value

Rd: Destination register

Rs: Source register

Figure 8: Format 3

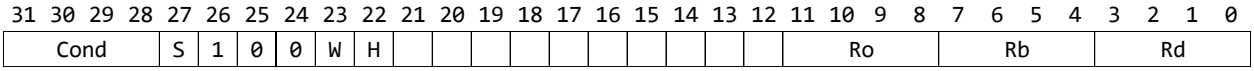
Operation

Instructions of this format perform a ternary or binary ALU operation on the given source register and immediate value and places the result in the destination register unless it's a comparison operation.

Examples

```
CMP   A, '\0'      ; Compares the ascii value in A to '\0' (0)
MOV   B, '\n'      ; Loads the ascii value for '\n' (10) into register B
ADD   B, 'A'       ; Adds the ascii value for 'A' (65) to register B
ADD   SP, 1        ; SP = SP + 1
CMP   B,  0        ; Subtract 0 from B and set the condition codes
AND   A,  B, 0b111      ; A = B & 0b111
MOV   A,  -1       ; Move the value -1 to A
```

Format 4: Load and Store



Cond: Condition code

S: Storing bit

W: Word bit

H: Half-word bit

Rd: Destination register

Rb: Base register

Ro: Offset register

Figure 9: Format 4

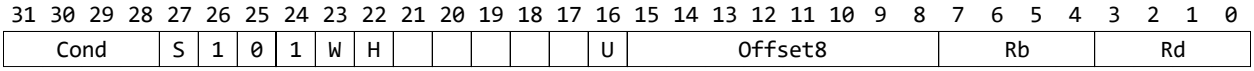
Operation

Instructions of this format transfer 32-bit values to and from registers and the memory space. The load or store operation depends on the S flag. The value in the offset register is added to the value in the base register and sent to the address BUS.

Examples

```
ST    [B, C], A   ; Stores the value in A to the address B + C
LD    A, [B, C]   ; Loads the value at address B + C into A
```

Format 5: Load and Store with Immediate Offset



Cond: Condition code

S: Storing bit

W: Word bit

H: Half-word bit

Rd: Destination register

Rb: Base register

U: Unsigned bit

Offset8: Immediate 8-bit offset

Figure 10: Format 5 with immediate offset

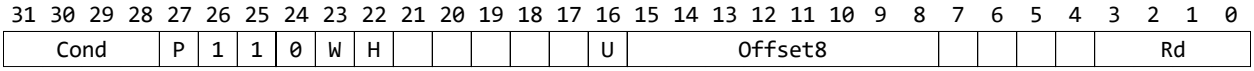
Operation

Instructions of this format transfer 32-bit values to and from registers and memory space. The load or store operation depends on the S flag. The immediate offset value is added to the value in the base register and sent to the address BUS.

Examples

```
ST    [SP, 2], C  ; Stores the value in C to the address SP + 2
LD    C, [SP, 2]  ; Loads the value at address SP + 2 into C
LD    A, [A]      ; Loads the value at the address in A into A
ST    [B], A      ; Stores the value in A to the address in B
```

Format 6: Push and Pop



Cond: Condition code

P: Pushing bit (1 = Pushing, 0 = popping)

W: Word bit

H: Half-word bit

Rd: Destination register

U: Unsigned bit

Offset8: Immediate 8-bit offset

Figure 11: Format 6

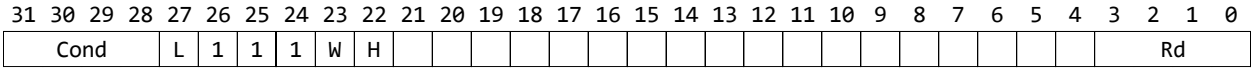
Operation

Instructions of this format transfer 32-bit values to and from registers and the stack in the RAM module. The push or pop operation depends on the P flag. The SP register is also altered depending on the operation.

Examples

```
PUSH  A     ; Pushes A to the stack and decrements SP
POP   A     ; Pops the top of the stack into A and increments SP
```

Format 7: Load Immediate



Cond: Condition code

L: Linking bit

W: Word bit

H: Half-word bit

Rd: Destination register

Figure 12: Format 7

Operation

Instructions of this format load a full 32-bit immediate value into one of the addressable registers. The 32-bit value is retrieved from the next location in memory.

Examples

```
LDI   A, 3000      ; Loads the value 3000 from memory into register A
LDI   B, =msg      ; Loads the label address into register B
LDI   C, 0x7fffffff     ; Loads the hex value 7fffffff into register C
```