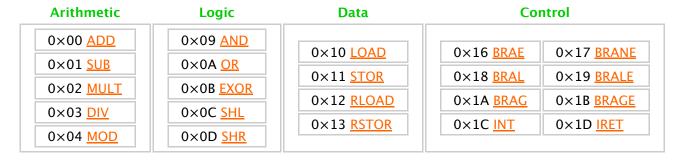
The Virtual Embedded Architectures Project

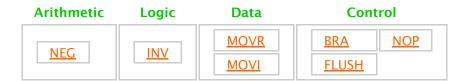
0×1 Architecture
0×1 Instruction Set
About the project
The Mini-A-Team
Coming soon

0×1 Instruction Set

Opcode table



Pseudoinstructions



Arithmetic instructions

ADD - Addition

```
General usage: ADD rA, rB, (rC + imm32)

Operation Detail: reg (rA) = reg (rB) + (reg (rC) + imm32);

Opcode: 0x00

Example Usage: ADD r6, r4, (104 + r5)

Example Encoding (bin):

Example Encoding (hex): 0x0006040500000068
```

SUB - Subtraction

```
General usage: SUB rA, rB, (rC + imm32)

Operation Detail: reg (rA) = reg (rB) - (reg (rC) + imm32);

Opcode: 0x01

Example Usage: SUB r3, r8, (15 + r5)

Example Encoding (bin):

Example Encoding (hex): 0x080308050000000F
```

MULT - Multiplication

```
General usage: MULT rA, rB, (rC + imm32)

Operation Detail: reg (rA) = reg (rB) * (reg (rC) + imm32);

Opcode: 0x02

Example Usage: MULT r13, r7, (30 + r2)

Example Encoding (bin):

Example Encoding (hex): 0x100D07020000001E
```

DIV - Division to find quotient

http://miniat.cs.sunyit.edu/?page_id=74 Page 2 of 13

```
General usage: DIV rA, rB, (rC + imm32)

Operation Detail: reg (rA) = reg (rB) / (reg (rC) + imm32);

Opcode: 0x03

Example Usage: DIV r8, r5, (14 + r7)

Example Encoding (bin): 0x180805070000000E

Example Encoding (hex): 0x180805070000000E
```

MOD - Modulus division to find remainder

```
General usage: MOD rA, rB, (rC + imm32)

Operation Detail: reg (rA) = reg (rB) % (reg (rC) + imm32);

Opcode: 0x04

Example Usage: MOD r10, r20, (17 + r30)

Example Encoding (bin):

Example Encoding (hex): 0x200A141E00000011
```

Logic instructions

AND - Bitwise AND

```
General usage: AND rA, rB, (rC + imm32)

Operation Detail: reg (rA) = reg (rB) & (reg (rC) + imm32);

Opcode: 0x09

Example Usage: AND r2, r3, (0x98765432 + r4)

Example Encoding (bin): 01001 ??P 00000010 00000011 00011000_01110110_01010100_00110010
```

http://miniat.cs.sunyit.edu/?page_id=74 Page 3 of 13

```
Example Encoding 0×4802030498765432 (hex):
```

OR - Bitwise OR

```
General usage: OR rA, rB, (rC + imm32)

Operation Detail: reg (rA) = reg (rB) | (reg (rC) + imm32);

Opcode: 0x0A

Example Usage: OR r5, r6, (0xBADCCODE + r7)

Example Encoding (bin):

Example Encoding (hex): 0x50050607BADCCODE
```

EXOR - Bitwise Exclusive OR

```
General usage: EXOR rA, rB, (rC + imm32)

Operation Detail: reg (rA) = reg (rB) ^ (reg (rC) + imm32);

Opcode: 0x0B

Example Usage: EXOR r8, r9, (r10 + 0xFEEDCAFE)

Example Encoding (bin):

Example Encoding (hex): 0x5808090AFEEDCAFE
```

SHL - Shift left

http://miniat.cs.sunyit.edu/?page_id=74 Page 4 of 13

Example Encoding 0×600B0C0DBA5EBA11 (hex):

SHR - **Shift** right

```
Description:
Logically shifts bits to the right a calculated number of positions, bringing 0 into opened high order bits.

Operation Detail: reg (rA) = reg (rB) >> (reg (rC) + imm32);
Opcode: 0x0D

Example Usage: SHR r14, r15, (r1 + 0xB01DFACE)

Example Encoding (bin):

Example Encoding (hex): 0x680E0F01B01DFACE
```

Data instructions

LOAD - Load a register from memory

```
Description:
Register A is loaded with a copy of the data at the effective address found by adding the contents of register C to an immediate.

Operation Detail: reg (rA) = mem (reg (rC) + imm32);
Opcode: 0×10

Example Usage: LOAD r2, [r3 + 4]

Example Encoding (bin):

Example Encoding (hex): 0×8002000300000004
```

http://miniat.cs.sunyit.edu/?page_id=74 Page 5 of 13

STORE - Store a register value in memory

```
Description: The data in register A is copied to the effective address found by adding the contents of register C to an immediate.

Operation Detail: mem (reg (rC) + imm32) = reg (rA);

Opcode: Ox11

Example Usage: STORE [r4 -1], r5

Example Encoding (bin): 0x88050004FFFFFFFF

Ox88050004FFFFFFFF
```

RLOAD - Load a register from an indexed register

```
Description:
Register A is loaded with a copy of the data in the register with the effective index found by adding the contents of register C to an immediate.

Operation Detail: reg (rA) = reg (reg (rC) + imm32);
Opcode: 0x12

Example Usage: RLOAD r2, [r3 + 4]

Example Encoding (bin):

Example Encoding (hex): 0x9002000300000004
```

RSTORE - Store a register value in an indexed register

```
General usage: RSTORE [rC + imm32], rA

Description: The data in register A is copied to the register with the effective index found by adding the contents of register C to an immediate.

Operation Detail: reg (reg (rC) + imm32) = reg (rA);

Opcode: 0x13

Example Usage: RSTORE [r4 -1], r5
```

http://miniat.cs.sunyit.edu/?page_id=74 Page 6 of 13

Control instructions

The following conditional branch instructions provide a hint to the hardware about the likelihood a branch will be taken. A braced value, "{true}" or "{false}", is given to indicate respectively whether the branch should be prefetched or not. This static hinting approach is obviously not the greatest branch prediction technique, but it will allow students debugging their applications to easily predict what will be fetched to the pipeline. When encoded, the Hint bit will be set to 1 if the hint is "{true}" and cleared to 0 for "{false}".

BRAE - Branch if equal

```
Description:

The Program Counter (PC) register is replaced by the effective address found by adding the contents of register C to an immediate only if the value in register A is equal to the value in register B.

Operation Detail:

if (reg (rA) == reg (rB)) {
    reg (PC) = reg (rC) + imm32;
}

Opcode:

Example Usage:

BRAE [r2 + 0x1ADA], r3, r4

Example Encoding (bin):

Example Encoding (hex):

Notes:
```

BRANE - Branch if not equal

General usage:

```
The Program Counter (PC) register is replaced by the effective address found by adding the contents of register C to an immediate only if the value in register A is not equal to the value in register B.

Operation Detail:

if (reg (rA) == reg (rB)) {
    reg (PC) = reg (rC) + imm32;
    }

Opcode:

Example Usage:

Example Encoding (bin):

Example Encoding (hex):

Notes:
```

BRAL - Branch if less than

BRALE - Branch if less than or equal

```
Description: The Program Counter (PC) register is replaced by the effective address found by adding the contents of register C to an immediate only if the value in register A is less than or equal to the value in register B.
```

BRAG - Branch if greater than

BRAGE - Branch if greater than or equal

```
General usage:
Description:
The Program Counter (PC) register is replaced by the effective address found by adding the contents of register C to an immediate only if the value in register A is greater than or equal to the value in register B.

Operation Detail: if (reg (rA) <= reg (rB)) {
    reg (PC) = reg (rC) + imm32;
    }

Opcode: 0x1B

Example Usage: BRAGE [r14 + 0x1BAA], r15, r1</pre>
```

```
Example Encoding (bin):

Example Encoding (hex):

OxD80F010E00001BAA
```

INT - Generate an interrupt

```
Description: Sets the Interrupt Flag Register bit for the interrupt number found by adding register C to an immediate.

Operation Detail: IFR = IFR | (1 << (reg (rc) + imm32));

Opcode: Ox1C

Example Usage: INT (r9 + 1)

Example Encoding (bin): 

Example Encoding (hex): OxE000000900000001
```

IRET - Return from an interrupt

Pseudoinstructions

http://miniat.cs.sunyit.edu/?page_id=74

The following pseudo instructions are defined soley to make programs more readable. However, these pseudo instructions are each replaced with a single core instruction; **NOT** a sequence of instructions.

NEG - Negation pseudoinstruction

INVERT - Inverts a register pseudoinstruction

```
Description: Invert all bits of the value contained in register B, storing the result into register A.

Operation Detail: EXOR rA, rB, (r0 + 0xffffffff)

Example Usage: INV r7, r11

Example Encoding (bin): 

Example Encoding (hex): 0x58070B00fffffffff
```

MOVR - Move register pseudoinstruction

http://miniat.cs.sunyit.edu/?page_id=74 Page 11 of 13

MOVI - Move immediate pseudoinstruction

BRA - Unconditional branch pseudoinstruction

```
Description:

The Program Counter (PC) register is replaced by the effective address found by adding the contents of register C to the immediate value.

Operation Detail:
Example Usage:

BRAE {T} [rC + imm32], r0, r0

Example Encoding (bin):

Example Encoding (bin):

Example Encoding (hex):

0×8200000202030000
```

NOP - No operation performed pseudoinstruction

http://miniat.cs.sunyit.edu/?page_id=74 Page 12 of 13

$\textbf{FLUSH-Empty the pipeline} \ \ ^{\text{pseudoinstruction}}$

General usage:	FLUSH
Description:	All pipeline stages are filled with NOPs. The PC queue is emptied, as well, and the address of the instruction scheduled after the flush is added back to the PC queue.
Operation Detail:	BRAE {F} [PC + 0], r0, r0
Example Usage:	FLUSH
Example Encoding (bin):	10000 ?0P 00000000 00000000 111111111 00000000_00000000
Example Encoding (hex):	0×800000FF00000000

http://miniat.cs.sunyit.edu/?page_id=74 Page 13 of 13

[&]quot;Even this could be a MiniAT peripheral."