Modules:

- **ALU**
  - Arithmetic_32bit

Operation	Code
add	0000
addu	0001
sub	0010
subu	0011

  - Logic_32bit

Operation	Code
and	0100
or	0101
sll	0110

  - Conditional_32bit

Operation	Code
slt	1000
sltu	1001

ALU Control

Operation	**ALUop**	**Funct**	**ALUctr**
add	100	10 0000	0000
addu	100	10 0001	0001
sub	100	10 0010	0010
subu	100	10 0011	0011
and	100	10 0100	0100
or	100	10 0101	0101
sll	100	00 0000	0110
slt	100	10 1010	1000
sltu	100	10 1011	1001
I_add	000	–	1010
I_sub	001	–	1011

bills_branch:

I-Address	Instruction (hex)	instruction (binary)	Instruction
00400020	20050001	001000-00000-00101-0000000000000001	addi-0-5-1
00400024	20060064	001000-00000-00110-0000000001100100	addi-0-6-100
00400028	20021000	001000-00000-00010-0001000000000000	addi-0-2-4096
0040002c	00421400	000000-00010-00010-00010-10000-000000	sll-2-2-2-16
00400030	20470028	001000-00010-00111-0000000000101000	addi-2-7-40
00400034	8c430000	100011-00010-00011-0000000000000000	lw-2-3-0
00400038	00c3202a	000000-00110-00011-00100-00000-101010	slt-6-3-4-0
0040003c	10850002	000100-00100-00101-0000000000000010	beq-4-5-2
00400040	00c33022	000000-00110-00011-00110-00000-100010	sub-6-3-6-0
00400044	ac400000	101011-00010-00000-0000000000000000	sw-2-0-0
00400048	20420004	001000-00010-00010-0000000000000100	addi-2-2-4
0040004c	1447fff9	000101-00010-00111-1111111111111001	bne-2-7-65529
00400050	ace60000	101011-00111-00110-0000000000000000	sw-7-6-0

D-Address	Data (hex)	Data
10000000	0000000a	10
10000004	00000009	9
10000008	00000008	8
1000000c	000002bc	700
10000010	00000005	5
10000014	00000006	6
10000018	00000190	400
1000001c	00000001	1
10000020	00000002	2
10000024	00000003	3

Program	Execution
addi: Reg[5] = Reg[0] + sgnext(+1)	Reg[5] = +1
addi: Reg[6] = Reg[0] + sgnext(+100)	Reg[6] = +100
addi: Reg[2] = Reg[0] + sgnext(+4096)	Reg[2] = +4096 = $(0\ 0\ 0\ 0\ 1\ 0\ 0\ 0)_{hex}$
sll: Reg[2] =Reg[2] << 16	R[2] = 2^{28} = $(1\ 0\ 0\ 0\ 0\ 0\ 0\ 0)_{hex}$
addi: Reg[7] = Reg[2]+sgnext(+40)	R[7] = $(1\ 0\ 0\ 0\ 0\ 0\ 2\ 8)_{hex}$
lw: Reg[3] = M[R[2]+sgnext(0)]	R[3] = M[$(1\ 0\ 0\ 0\ 0\ 0\ 0\ 0)_{hex}$ = 10
slt: Reg[4] = (Reg[6] < Reg[3]) ?	R[4] = (+100 < 10) = $(0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)_{hex}$
beq: if (Reg[4] == Reg[5]) PC = PC + 4 + 2	R[4] ! = R[5] no branch
sub: Reg[6] = Reg[6] - Reg[3]	R[6] = +100 - 10 = 90 = $(0\ 0\ 0\ 0\ 0\ 0\ 5\ A)_{hex}$
sw: M[Reg[2] + sgnext(0)] = Reg[0]	M[$(1\ 0\ 0\ 0\ 0\ 0\ 0\ 0)_{hex}$ = 0
addi: Reg[2] = Reg[2] + sgnext(+4)	R[2] = $(1\ 0\ 0\ 0\ 0\ 0\ 0\ 4)_{hex}$
bne: if (Reg[2] != Reg[7]) PC=PC+4+65529	PC = (0040004c) + 4 + (ffffffe4) = (0040 0034) $\rightarrow$addi
lw: Reg[3] = M[R[2]+sgnext(0)]	R[3] = M[$(1\ 0\ 0\ 0\ 0\ 0\ 0\ 4)_{hex}$ = 9
slt: Reg[4] = (Reg[6] < Reg[3]) ?	R[4] = (+100 < 9) = $(0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)_{hex}$
beq: if (Reg[4] == Reg[5]) PC = PC + 4 + 2	R[4] ! = R[5] no branch
sub: Reg[6] = Reg[6] - Reg[3]	R[6] = 90 - 9 = 81 = $(0000\ 0051)_{hex}$
sw: M[Reg[2] + sgnext(0)] = Reg[0]	M[$(1\ 0\ 0\ 0\ 0\ 0\ 0\ 4)_{hex}$ = 0
addi: Reg[2] = Reg[2] + sgnext(+4)	R[2] = $(1\ 0\ 0\ 0\ 0\ 0\ 0\ 8)_{hex}$
bne: if (Reg[2] != Reg[7]) PC=PC+4+65529	PC = (0040004c) + 4 + (ffffffe4) = (0040 0034) $\rightarrow$addi
lw: Reg[3] = M[R[2]+sgnext(0)]	R[3] = M[$(1\ 0\ 0\ 0\ 0\ 0\ 0\ 8)_{hex}$ = 8
slt: Reg[4] = (Reg[6] < Reg[3]) ?	R[4] = (+100 < 8) = $(0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)_{hex}$
beq: if (Reg[4] == Reg[5]) PC = PC + 4 + 2	R[4] ! = R[5] no branch
sub: Reg[6] = Reg[6] - Reg[3]	R[6] = 81 - 8 = 73 = $(0000\ 0049)_{hex}$
sw: M[Reg[2] + sgnext(0)] = Reg[0]	M[$(1\ 0\ 0\ 0\ 0\ 0\ 0\ 8)_{hex}$] = 0
addi: Reg[2] = Reg[2] + sgnext(+4)	R[2] = $(1\ 0\ 0\ 0\ 0\ 0\ 0\ c)_{hex}$
bne: if (Reg[2] != Reg[7]) PC=PC+4+65529	PC = (0040004c) + 4 + (ffffffe4) = (0040 0030) $\rightarrow$addi
lw: Reg[3] = M[R[2]+sgnext(0)]	R[3] = M[$(1\ 0\ 0\ 0\ 0\ 0\ 0\ c)_{hex}$ = 700
slt: Reg[4] = (Reg[6] < Reg[3]) ?	R[4] = (+100 < 700) = $(0\ 0\ 0\ 0\ 0\ 0\ 0\ 1)_{hex}$
beq: if (Reg[4] == Reg[5]) PC = PC + 4 + 2	PC =(0040003c) + 4 + 8 = (00400048)
addi: Reg[2] = Reg[2] + sgnext(+4)	R[2] = $(1\ 0\ 0\ 0\ 0\ 0\ 1\ 0)_{hex}$
bne: if (Reg[2] != Reg[7]) PC=PC+4+65529	PC = (0040004c) + 4 + (ffffffe4) = (0040 0030) $\rightarrow$addi

lw: Reg[3] = M[R[2]+sgnext(0)]	R[3] = M[$(1\ 0\ 0\ 0\ 0\ 0\ 1\ 0)_{hex}$ = 5
slt: Reg[4] = (Reg[6] < Reg[3]) ?	R[4] = (+100 < 5) = $(0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)_{hex}$
beq: if (Reg[4] == Reg[5]) PC = PC + 4 + 2	R[4] ! = R[5] no branch
sub: Reg[6] = Reg[6] - Reg[3]	R[6] = 73 - 5 = 68 = $(0000\ 0044)_{hex}$
sw: M[Reg[2] + sgnext(0)] = Reg[0]	M[$(1\ 0\ 0\ 0\ 0\ 0\ 1\ 0)_{hex}$] = 0
addi: Reg[2] = Reg[2] + sgnext(+4)	R[2] = $(1\ 0\ 0\ 0\ 0\ 0\ 1\ 4)_{hex}$
bne: if (Reg[2] != Reg[7]) PC=PC+4+65529	PC = (0040004c) + 4 + (ffffffe4) = (0040 0030) $\rightarrow$addi

lw: Reg[3] = M[R[2]+sgnext(0)]	R[3] = M[$(1\ 0\ 0\ 0\ 0\ 0\ 1\ 0)_{hex}$] = 6
slt: Reg[4] = (Reg[6] < Reg[3]) ?	R[4] = (+100 < 6) = $(0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)_{hex}$
beq: if (Reg[4] == Reg[5]) PC = PC + 4 + 2	R[4] ! = R[5] no branch
sub: Reg[6] = Reg[6] - Reg[3]	R[6] = 68 - 6 = 62 = $(0000\ 003e)_{hex}$
sw: M[Reg[2] + sgnext(0)] = Reg[0]	M[$(1\ 0\ 0\ 0\ 0\ 0\ 1\ 4)_{hex}$ = 0
addi: Reg[2] = Reg[2] + sgnext(+4)	R[2] = $(1\ 0\ 0\ 0\ 0\ 0\ 1\ 8)_{hex}$
bne: if (Reg[2] != Reg[7]) PC=PC+4+65529	PC = (0040004c) + 4 + (ffffffe4) = (0040 0030) $\rightarrow$addi

lw: Reg[3] = M[R[2]+sgnext(0)]	R[3] = M[$(1\ 0\ 0\ 0\ 0\ 0\ 1\ 8)_{hex}$ = 400
slt: Reg[4] = (Reg[6] < Reg[3]) ?	R[4] = (+100 < 400) = $(0\ 0\ 0\ 0\ 0\ 0\ 0\ 1)_{hex}$
beq: if (Reg[4] == Reg[5]) PC = PC + 4 + 2	PC =(0040003c) + 4 + 8 = (00400048)
addi: Reg[2] = Reg[2] + sgnext(+4)	R[2] = $(1\ 0\ 0\ 0\ 0\ 0\ 1\ c)_{hex}$
bne: if (Reg[2] != Reg[7]) PC=PC+4+65529	PC = (0040004c) + 4 + (ffffffe4) = (0040 0030) $\rightarrow$addi

lw: Reg[3] = M[R[2]+sgnext(0)]	R[3] = M[$(1\ 0\ 0\ 0\ 0\ 0\ 1\ c)_{hex}$ = 1
slt: Reg[4] = (Reg[6] < Reg[3]) ?	R[4] = (+100 < 1) = $(0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)_{hex}$
beq: if (Reg[4] == Reg[5]) PC = PC + 4 + 2	R[4] ! = R[5] no branch
sub: Reg[6] = Reg[6] - Reg[3]	R[6] = 62 - 1 = 61 = $(0000\ 003d)_{hex}$
sw: M[Reg[2] + sgnext(0)] = Reg[0]	M[$(1\ 0\ 0\ 0\ 0\ 0\ 1\ c)_{hex}$ = 0
addi: Reg[2] = Reg[2] + sgnext(+4)	R[2] = $(1\ 0\ 0\ 0\ 0\ 0\ 2\ 0)_{hex}$
bne: if (Reg[2] != Reg[7]) PC=PC+4+65529	PC = (0040004c) + 4 + (ffffffe4) = (0040 0030) $\rightarrow$addi

lw: Reg[3] = M[R[2]+sgnext(0)]	R[3] = M[$(1\ 0\ 0\ 0\ 0\ 0\ 2\ 0)_{hex}$ = 2
slt: Reg[4] = (Reg[6] < Reg[3]) ?	R[4] = (+100 < 2) = $(0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)_{hex}$
beq: if (Reg[4] == Reg[5]) PC = PC + 4 + 2	R[4] ! = R[5] no branch
sub: Reg[6] = Reg[6] - Reg[3]	R[6] = 61 - 2 = 59 = $(0000\ 003b)_{hex}$
sw: M[Reg[2] + sgnext(0)] = Reg[0]	M[$(1\ 0\ 0\ 0\ 0\ 0\ 1\ c)_{hex}$] = 0
addi: Reg[2] = Reg[2] + sgnext(+4)	R[2] = $(1\ 0\ 0\ 0\ 0\ 0\ 2\ 4)_{hex}$
bne: if (Reg[2] != Reg[7]) PC=PC+4+65529	PC = (0040004c) + 4 + (ffffffe4) = (0040 0030) $\rightarrow$addi

lw: Reg[3] = M[R[2]+sgnext(0)]	R[3] = M[$(1\ 0\ 0\ 0\ 0\ 0\ 1\ 0)_{hex}$ = 3
slt: Reg[4] = (Reg[6] $<$ Reg[3]) ?	R[4] = (+100 $<$ 3) = $(0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)_{hex}$
beq: if (Reg[4] == Reg[5]) PC = PC + 4 + 2	R[4] ! = R[5] no branch
sub: Reg[6] = Reg[6] - Reg[3]	R[6] = 59 - 3 = 56 = $(0000\ 0038)_{hex}$
sw: M[Reg[2] + sgnext(0)] = Reg[0]	M[$(1\ 0\ 0\ 0\ 0\ 0\ 1\ 0)_{hex}$ = 0
addi: Reg[2] = Reg[2] + sgnext(+4)	R[2] = $(1\ 0\ 0\ 0\ 0\ 0\ 2\ 8)_{hex}$
bne: if (Reg[2] != Reg[7]) PC=PC+4+65529	no branch $\rightarrow$addi
sw: M[Reg[7] + sgnext(0)] = Reg[6]	R[7] = 56 = (0000 0038)

sort_corrected_branch:

Address	Instruction (hex)	instruction (binary)	Instruction
00400020	20021000	001000-00000-00010-0001000000000000	addi-0-2-4096
00400024	00421400	000000-00010-00010-00010-10000-000000	sll-2-2-2-16
00400028	20440024	001000-00010-00100-0000000000100100	addi-2-4-36
0040002c	20450028	001000-00010-00101-0000000000101000	addi-2-5-40
00400030	8c470000	100011-00010-00111-0000000000000000	lw-2-7-0
00400034	20430004	001000-00010-00011-0000000000000100	addi-2-3-4
00400038	8c610000	100011-00011-00001-0000000000000000	lw-3-1-0
0040003c	00e1302a	000000-00111-00001-00110-00000-101010	slt-7-1-6-0
00400040	1cc00003	000111-00110-00000-0000000000000011	bgtz-6-0 3
00400044	ac410000	101011-00010-00001-0000000000000000	sw-2-1-0
00400048	ac670000	101011-00011-00111-0000000000000000	sw-3-7-0
0040004c	00203820	000000-00001-00000-00111-00000-100000	add-1-0-7-0
00400050	20630004	001000-00011-00011-0000000000000100	addi-3-3-4
00400054	1465fff8	000101-00011-00101-1111111111111000	bne-3-5-65529
00400058	20420004	001000-00010-00010-0000000000000100	addi-2-2-4
0040005c	1444fff4	000101-00010-00100-1111111111110100	bne-2-4-65525

D-Address	Data (hex)	Data
10000000	00000009	9
10000004	0000000a	10
10000008	00000008	8
1000000c	00000007	7
10000010	00000005	5
10000014	00000006	6
10000018	00000004	4
1000001c	00000001	1
10000020	00000002	2
10000024	00000003	3

Program:

addi: Reg[2]=Reg[0] + sgnext(+4096)
sll: Reg[2] = Reg[2] << 16
addi: Reg[4] = Reg[2] + sgnext(+36)
addi: Reg[5] = Reg[2] + sgnext(+40)
lw: Reg[7] = M[Reg[2] + sgnext(0)]
addi: Reg[3] = Reg[2] + sgnext(4)
lw: Reg[1] = M[Reg[3] + sgnext(0)]
slt: Reg[6] = (Reg[7] < Reg[1])
bgtz(?): if (R[6]¿R[0]) PC = PC + 4 + c
sw: M[Reg[2] + sgnext(0)] = Reg[1]
sw: M[Reg[3] + sgnext(0)] = Reg[7]
add: Reg[7] = Reg[1] + Reg[0]

```
   addi: Reg[3] = Reg[3] + sgnext(+4)
   bne: if (R[3] ! = R[5]) PC=PC+4+65529
   addi: Reg[2] = Reg[2]+sgnext(+4)
   bne: if (R[2] ! = R[4]) PC=PC+4+65525
```

Program	Execution
addi: Reg[2]=Reg[0] + sgnext(+4096)	Reg[2] = (0000 1000)
sll: Reg[2] = Reg[2] << 16	Reg[2] = (1000 0000)
addi: Reg[4] = Reg[2] + sgnext(+36)	Reg[4]= (1000 0024)
addi: Reg[5] = Reg[2] + sgnext(+40)	Reg[5] = (1000 0028)
lw: Reg[7] = M[Reg[2] + sgnext(0)]	Reg[7] = 9
addi: Reg[3] = Reg[2] + sgnext(4)	Reg[3] = (1000 0004)
lw: Reg[1] = M[Reg[3] + sgnext(0)]	Reg[1] = 10
slt: Reg[6] = (Reg[7] < Reg[1])	Reg[6] = (9 < 10) = (00000001)
bgtz(?): if (R[6]>R[0]) PC = PC + 4 + c	PC = (00400040) + 4 + c = (0040 0050)
addi: Reg[3] = Reg[3] + sgnext(+4)	Reg[3] = (1000 0008)
bne: if (R[3] ! = R[5]) PC=PC+4+65529	PC = (0040 0054) + 4 +

unsigned_sum:

Address	Instruction (hex)	instruction (binary)	Instruction
00400020	00002820	000000-00000-00000-00101-00000-100000	add-0-0-5-0
00400024	20071000	001000-00000-00111-0001000000000000	addi-0-7-4096
00400028	00e73c00	000000-00111-00111-00111-10000-000000	sll-7-7-7-16
0040002c	00e03020	000000-00111-00000-00110-00000-100000	add-7-0-6-0
00400030	20c60028	001000-00110-00110-0000000000101000	addi-6-6-40
00400034	8ce40000	100011-00111-00100-0000000000000000	lw-7-4-0
00400038	00a42821	000000-00101-00100-00101-00000-100001	addu-5-4-5-0
0040003c	20e70004	001000-00111-00111-0000000000000100	addi-7-7-4
00400040	14e6fffc	000101-00111-00110-1111111111111100	bne-7-6-65532
00400044	ace50000	101011-00111-00101-0000000000000000	sw-7-5-0

D-Address	Data (hex)	Data
10000000	0000000f	15
10000004	000000f0	240
10000008	00000f00	3840
1000000c	0000f000	61440
10000010	000f0000	983040
10000014	00f00000	15728640
10000018	0f000000	251658240
1000001c	10000000	268435456
10000020	20000000	536870912
10000024	c0000000	3.221210^9
10000028	ffffffff	4.295010^9

Program:

add: Reg[5] = Reg[0]+Reg[0]
addi: Reg[7] = Reg[0] + sgnext(+4096)
sll: Reg[7] = Reg[7] << 16
add: Reg[6] = Reg[7] + Reg[0]
addi: Reg[6] = Reg[6] + sgnext(+40)
lw: Reg[4] = M[Reg[7] + sgnext(0)]
addu: Reg[5] = Reg[5] + Reg[4]
addi: Reg[7] = Reg[7] + sgnext(+4)
bne: if (Reg[7] ! = Reg[6]) PC = PC + 4 + 65532
sw: M[Reg[7] + sgnext(0)] = Reg[5]

Program	Execution
add: Reg[5] = Reg[0]+Reg[0]	R[5] =0
addi: Reg[7] = Reg[0] + sgnext(+4096)	R[7] = 4096 = (00001000)
sll: Reg[7] = Reg[7] << 16	R[7] = (1000 0000)
add: Reg[6] = Reg[7] + Reg[0]	R[6] = (1000 0000)
addi: Reg[6] = Reg[6] + sgnext(+40)	R[6]=(1000 0028)
lw: Reg[4] = M[Reg[7] + sgnext(0)]	R[4] = M[(1000 0000))] = 15
addu: Reg[5] = Reg[5] + Reg[4]	R[5] = 0 + 15 = 15
addi: Reg[7] = Reg[7] + sgnext(+4)	R[7] = (1000 0004)
bne: if (Reg[7] ! = Reg[6]) PC = PC + 4 + 65532	PC = (00400040)+4+(ffff fffc) = (00400040)

bne jumps back to itself – gets stuck!