

CSE3666 HW3 solutions

1. Function.

foo:

save ra, s1, s2, s3, and s4 on stack

addi sp, sp, -20

sw ra, 16(sp)

sw s4, 12(sp)

sw s3, 8(sp)

sw s2, 4(sp)

sw s1, 0(sp)

addi s4, a0, 0 # save d in s4

addi s3, a1, 0 # copy n to s3

addi s1, x0, 0 # sum = 0

addi s2, x0, 0 # i = 0

beq x0, x0, test

loop:

call bar(&d[i], n - i);

slli a0, s2, 2 # 4 * i

add a0, a0, s4 # d[i]'s address, the 1st argument

sub a1, s3, s2 # n - i, the 2nd argument

jal ra, bar

add s1, s1, a0 # sum += return value

loop control

addi s2, s2, 1 # i += 1

test:

blt s2, s3, loop

set return value

addi a0, s1, 0 # move return value to a0

restore ra, s1, s2, s3, and s4

lw ra, 16(sp)

lw s4, 12(sp)

lw s3, 8(sp)

lw s2, 4(sp)

lw s1, 0(sp)

addi sp, sp, 20

jalr x0, ra, 0

2. Function.

msort:

```
# save ra, s1, s2, and s3
addi sp, sp, -16
sw    ra, 12(sp)
sw    s3, 8(sp)
sw    s2, 4(sp)
sw    s1, 0(sp)
addi  sp, sp, -1024    # allocate space for array c

addi  t0, t0, 1        # n <= 1?
blt   t0, a1, continue
beq   x0, x0, Exit
```

Continue:

```
addi  s1, a0, 0        # save d in s1
addi  s2, a1, 0        # save n in s2
srai  s3, s2, 1        # n1 = n/2

# msort(d, n1);
addi  a1, s3, 0        # n1. a0 is already set
jal   ra, msort
# msort(&d[n1], n - n1);
slli  a0, s3, 2        # n1 * 4
add   a0, a0, s1        # d[n1]'s address
sub   a1, s2, s3       # n - n1
jal   ra, msort
# merge(c, d, n1, &d[n1], n - n1);
addi  a0, sp, 0        # c
addi  a1, s1, 0        # d
addi  a2, s3, 0        # n1
slli  a3, s3, 2        # d[n1]'s address
add   a3, a3, s1
sub   a4, s2, s3       # n - n1
jal   ra, merge
# copy(d, c, n);
addi  a0, s1, 0
addi  a1, sp, 0
addi  a2, s2, 0
jal   ra, copy
```

Exit:

```
addi  sp, sp, 1024    # free space used by c
lw    ra, 12(sp)      # restore registers
lw    s3, 8(sp)
lw    s2, 4(sp)
lw    s1, 0(sp)
addi  sp, sp, 16
jalr  x0, ra, 0
```

3. Encoding branch and jal.

I10:

The offset is $(100-10) * 4 = 360$. The lower 13 bits are .

SB-type. The immediate is stored in funct7 and rd. 0000101101000

opcode: 1100011

rd: 01000

funct3: 101

rs1: 01010

rs2: 10100

funct7: 0001011

Machine code in bin: 00010111010001010101010001100011

Machine code in hex: 17455463

I11:

The offset is $(1-11) * 4 = -40$. The lower 13 bits are 1111111011000.

SB-type. The immediate is stored in funct7 and rd.

opcode: 1100011

rd: 11001

funct3: 000

rs1: 01010

rs2: 00000

funct7: 1111110

Machine code in bin: 11111100000001010000110011100011

Machine code in hex: FC050CE3

I140:

The offset is $(100-140) * 4 = -160$. We have -40 in I11.

The lower 21 bits are 1_1111_1111_1111_0110_0000.

The 20 bits in the immediate fields are

11110110000111111111

rd: 00000

opcode: 1101111

Machine code in bits: 11110110000111111111_00000_1101111

Machine code in hex: F61FF06F

4. Decoding branch and jal

```
0x0400366C:          0xDB5A04E3
11011011010110100000010011100011
opcode:   1100011
SB-type
funct3:   000
rs1:      10100
rs2:      10101
Instr[funct7,rd]:  1101101 01001
imm[12:0]:  110110101000
imm in decimal: -600
imm in hexadecimal: FFFFFFFDA8
The target address is 0x0400366C + 0xFFFFFFDA8 = 0x04003414
```

```
0x04208888:          0xFA9FF0EF
11111010100111111111000011101111
opcode:   1101111
UJ type. JAL
rd:       00001
Instr[31:12]: 111101010011111111
imm[20:0]:  1111111111110101000
imm in decimal: -88
imm in hexadecimal: FFFFFFFA8
The target address is 0x04208888 + 0xFFFFFA8 = 0x04208830
```