BÀI TẬP CHƯƠNG 2: ASSEMBLY MIPS VÀ MÃ MÁY

DẠNG 1: CHUYỂN ĐỔI GIỮA MÃ MÁY VÀ ASSEMBLY

Bài tập 1: Chuyển đổi Assembly MIPS sang mã máy (Hex)

a) add $t1, $t2, $t3

000000 01010 01011 01001 00000 100000 → 0x014B4820

b) sub $t0, $s0, $s1

000000 10000 10001 01000 00000 010010 → 0x02114012

c) addi $s1, $s2, 20

001000 10001 10010 0000000000010100 → 0x22320014

d) lw $s0, 24($t1)

100011 01001 10000 0000000000011000 → 0x8D300018

e) sw $s3, 40($t2)

101011 01010 10011 0000000000101000 → 0xAD530028

f) lw $t2, -8($s0)

100011 10000 01010 1111111111111000 → 0x8E0AFFF8

g) sh $s2, 12($t4)

101001 01100 10010 0000000000001100 → 0xA592000C

h) mult $t5, $t6

000000 01101 01110 00000 00000 011000 → 0x01AE0018

i) mfhi $s5

000000 00000 00000 10101 00000 010000 → 0x0000A810

j) sra $t1, $t0, 2

000000 00000 01000 01001 00010 000011 → 0x00084883

k) lui $s6, 12

001111 00000 10110 0000000000001100 → 0x3C16000C

l) j Label (Label = 0x00400020)

0x00400020 = 0000 0000 0100 0000 0000 0000 0010 0000

PC = Label >> 2 = 0000 0000 0001 0000 0000 0000 0000 1000

→ 000010 00000100000000000000001000 → 0x08100008

m) bne $t1, $t2, Label (Label = 0x00400008, PC = 0x00400004)

Label = PC + 4 + 4*offset → offset = 0

000101 01001 01010 0000000000000000 → 0x152A0000

n) start: addi $t0, $zero, 5

lw $t1, 12($s0)

sub $s1, $t0, $t2

beq $s1, $t0, end (1) (PC = 0x0040000C)

sw $t2, 20($s0)

END: ori $t3, $zero, 15 (Label = 0x00400014)

Chuyển lệnh (1) sang mã máy

Giả sử địa chỉ lệnh đầu tiên PC = 0x00400000

Label = PC + 4 + 4*offset → offset = 1

→ 000100 10001 01000 0000000000000001 → 0x12280001

o) Loop: addi $s2, $s2, 2 (Label = 0x004000000)

sub $s3, $s3, $s4

beq $s2, $s1, Loop (2) (PC = 0x004000008)

ori $s5, $zero, 12

Chuyển lệnh (2) sang mã máy

Giả sử địa chỉ lệnh đầu tiên PC = 0x00400000

Label = PC + 4 + 4*offset → offset = -3

→ 000100 10010 10001 1111111111111101 → 0x1251FFFD

Bài tập 2: Chuyển mã máy (Hex) sang assembly MIPS

a) 0xAD310018

→ 101011 01001 10001 0000000000011000 → sw $s1, 24($t1)

b) 0x8D28000C

→ 100011 01001 01000 0000000000001100 → lw $t0, 12($t1)

c) 0x0170001A

→ 000000 01011 10000 00000 00000 011010 → div $t3, $s0

d) 0x02325820

→ 000000 10001 10010 01011 00000 100000 → add $t3, $s1, $s2

e) 0x02538820

→ 000000 10010 10011 10001 00000 100000 → add $s1, $s2, $s3

f) 0x001090C3

→ 000000 00000 10000 10010 00011 000011 → sra $s2, $s0, 3

g) 0x014B4822

→ 000000 01010 01011 01001 00000 100010 → sub $t1, $t2, $t3

h) 0x85680004

i) 0x2151fff8

→ 001000 01010 10001 1111111111111000 → addi $s1, $t2, -8

j) 0x2109FFFC

→ 001000 01000 01001 1111111111111100 → addi $t1, $t0, -4

k) 0x8e28fff0

→ 100011 10001 01000 1111111111110000 → lw $t0, -16($s1)

l) 0xAE28FFFC

→ 101011 10001 01000 1111111111111100 → sw $t0, -4($s1)

m) 0x01534822

→ 000000 01010 10011 01001 00000 100010 → sub $t1, $t2, $s3

n) 0x08100009 (PC = 0x90400018)

→ 000010 00000100000000000000001001

PC + 4 = 0x9040001C

Address = 00000100000000000000001001

Label = Address << 2 = 0000 0000 0100 0000 0000 0000 0010 0100 → 0x00400024

→ j Label (Label = 0x90400024)

o) 0x1109fffd

→ 000100 01000 01001 1111111111111101 → beq $t0, $t1, Label (offset = -3)

p) 0x1632fff9

→ 000101 10001 10010 1111111111111001 → bne $s1, $s2, Label (offset = -7)

DẠNG 2: CHUYỂN TỪ C SANG ASSEMBLY MIPS VÀ NGƯỢC LẠI

Giả sử f, g, h, i, j = $s0, $s1, $s2, $s3, $s4; A, B = $s6, $s7

a, b, c, d = $a0, $a1, $a2, $a3

Bài tập 1: Chuyển từ C sang Assembly MIPS (biết mỗi word có 4 bytes)

a) B[5] = A[i] - A[j];

```
sll $t0, $s3, 2
add $t0, $s6, $t0
lw $t1, 0($t0)

sll $t2, $s4, 2
add $t2, $s6, $t2
lw $t3, 0($t2)

sub $t4, $t1, $t3

addi $t5, $s7, 20
sw $t4, 0($t5)
```

b) B[j] = A[i] + B[2];

```
sll $t0, $s3, 2
add $t0, $s6, $t0
lw $t1, 0($t0)

addi $t2, $s7, 8
lw $t3, 0($t2)

add $t4, $t1, $t3

sll $t5, $s4, 2
add $t5, $s7, $t5
sw $t4, 0($t5)
```

c) A[7] = B[i + j];

```
add $t0, $s3, $s4
sll $t0, $t0, 2
add $t0, $s7, $t0
lw $t1, 0($t0)

addi $t2, $s6, 28
sw $t1, 0($t2)
```

d) i = 5;

```
if (i < j) f = g - h;
else f = g + h;
ori $s3, $zero, 5;
slt $t0, $s3, $s4
beq $t0, $zero, ELSE
sub $s0, $s1, $s2
j END
ELSE: add $s0, $s1, $s2
END:
```

```
e) i = j + 2;
if (i == g) f = h + g;
addi $s3, $s4, 2
beq $s3, $s1, THEN
j END
THEN: add $s0, $s2, $s1
END:
```

```
f) i = 0;
while (i < 5) {
f = f + g;
i = i + 1;
}
ori $s3, $zero, 0
LOOP: slti $t0, $s3, 5
beq $t0, $zero, END
add $s0, $s0, $s1
addi $s3, $s3, 1
j LOOP
END:
```

```
g)
for (i = 0; i < 3; i++)
    f = f + h;
ori $s3, $zero, 0
LOOP: slti $t0, $s3, 3
beq $t0, $zero, END
add $s0, $s0, $s2
addi $s3, $s3, 1
j LOOP
END:
```

```
h) for (i = 0; i < j; i++)
f = f * g;
ori $s3, $zero, 0
LOOP: slt $t0, $s3, $s4
beq $t0, $zero, END
```

```
mul $s0, $s0, $s1
addi $s3, $s3, 1
j LOOP
END:
```

```
i) i = 0;
while (i < 10) {
if (i % 2 == 0) f = f + g;
else f = f - h;
i++;
}
ori $s3, $zero, 0
LOOP: slti $t0, $s3, 10
beq $t0, $zero, END

andi $t1, $s3, 1
bne $t1, $zero, ELSE
add $s0, $s0, $s1
j UPDATE
ELSE: sub $s0, $s0, $s2

UPDATE:
addi $s3, $s3, 1
j LOOP
END:
```

```
j) i = 5;
while (i < 15) {
f = g << 2;
h = h + f;
i++;
}
ori $s3, $zero, 5
LOOP: slti $t0, $s3, 15
beq $t0, $zero, END
sll $s0, $s1, 2
add $s2, $s2, $s0
addi $s3, $s3, 1
j LOOP
END:
```

```
k) i = 0;
do {
f = f + g;
i++;
} while (i < j);
ori $s3, $zero, 0
```

```
LOOP: add $s0, $s0, $s1
addi $s3, $s3, 1
slt $t0, $s3, $s4
bne $t0, $zero, LOOP
END:
```

l) A[3] = f + g;
if (A[3] > h) B[2] = A[3] - h;

```
addi $t0, $s6, 12
add $t1, $s0, $s1
sw $t1, 0($t0)

lw $t2, 0($t0)
slt $t3, $s2, $t2
beq $t3, $zero, END

addi $t4, $s7, 8
sub $t5, $t2, $s2
sw $t5, 0($t4)
END:
```

m) B[5] = A[i] + A[j];
if (B[5] < h) f = g + h;
else f = g - h;

```
sll $t0, $s3, 2
add $t0, $s6, $t0
lw $t1, 0($t0)

sll $t2, $s4, 2
add $t2, $s6, $t2
lw $t3, 0($t2)

add $t4, $t1, $t3
addi $t5, $s7, 20
sw $t4, 0($t5)

slt $t6, $t4, $s2
beq $t6, $zero, ELSE
add $s0, $s1, $s2
j END
ELSE: sub $s0, $s1, $s2
END:
```

o) i = 5;
do {
i = i - 2;
} while (i > 0);

```
ori $s3, $zero, 5
LOOP: addi $s3, $s3, -2
slt $t0, $zero, $s3
bne $t0, $zero, LOOP
END:
```

p) if (A[i] == g) f = g + h;
else f = g - h;

```
sll $t0, $s3, 2
add $t0, $s6, $t0
lw $t1, 0($t0)

beq $t1, $s1, THEN
sub $s0, $s1, $s2
j END
THEN: add $s0, $s1, $s2
END:
```

q) if (i >= j) f = A[j] - g;
else f = g | h;

```
slt $t0, $s3, $s4
beq $t0, $zero, THEN
or $s0, $s1, $s2
j END
THEN:
sll $t1, $s4, 2
add $t1, $s6, $t1
lw $t2, 0($t1)
sub $s0, $t2, $s1
END:
```

r) if (i > 0 && i < 10) f = g + h;
else f = g - h;

```
slt $t0, $zero, $s3
beq $t0, $zero, ELSE
slti $t1, $s3, 10
beq $t1, $zero, ELSE
add $s0, $s1, $s2
j END
ELSE: sub $s0, $s1, $s2
END:
```

s) if (i < 5 || i > 20) f = g & h;
else f = g | h;

```
slti $t0, $s3, 5
bne $t0, $zero, THEN
slti $t1, $s3, 21
```

```
beq $t1, $zero, THEN
or $s0, $s1, $s2
j END
THEN: and $s0, $s1, $s2
END:
```

```
t) int tinh(int a, int b) {
return a * b;
}
int f = tinh(2, 3);
ori $a0, $zero, 2
ori $a1, $zero, 3
jal tinh
addi $s0, $v0, 0

tinh:
mul $v0, $a0, $a1
jr  $ra
END:
```

```
u) int tong(int a, int b) {
return a + b;
}
f = tong(i, j) + tong(i, 4);
addi $a0, $s3, 0
addi $a1, $s4, 0
jal tong
addi $t0, $v0, 0

addi $a0, $s3, 0
ori $a1, $zero, 4
jal tong
add $s0, $t0, $v0

tong:
add $v0, $a0, $a1
jr $ra
END:
```

```
v) int tong(int a, int b) {
return a + b;
}
int func(int a, int b, int c, int d) {
return tong(a, b) - tong(c, d);
}
addi $a0, $a0, 0
addi $a1, $a1, 0
```

```
jal tong
addi $t0, $v0, 0

addi $a0, $a2, 0
addi $a1, $a3, 0
jal tong
sub $v0, $t0, $v0
jr $ra

tong:
add $v0, $a0, $a1
jr $ra
END:
```

```
w) int is_between(int a, int b, int c) {
return (a <= b && b <= c);
}
```

```
is_between:
slt $t0, $a1, $a0
bne $t0, $zero, FALSE
slt $t1, $a2, $a1
bne $t1, $zero, FALSE
ori $v0, $zero, 1
jr $ra
FALSE:
ori $v0, $zero, 0
jr $ra
END:
```

```
x) int tong(int a, int b) {
return a + b;
}
int func(int a, int b, int c, int d) {
return tong(a, b) + tong(c, d);
}
```

```
tong:
add $v0, $a0, $a1
jr $ra

func:
jal tong
ori $s0, $zero, 0
add $s0, $s0, $v0
ori $a0, $a2, 0
ori $a1, $a3, 0
jal tong
add $v0, $v0, $s0
```

```
jr $ra
```

y)

```
int complex_func(int a, int b, int c, int d) {
int sum1 = a + b;
int sum2 = c + d;
return sum1 * sum2;
}
complex_func:
add $t0, $a0, $a1
add $t1, $a2, $a3
mul $v0, $t0, $t1
jr $ra
```

```
z) int TowerHanoi(int N){
if(N <= 0) return 0;
else if(N == 1) return 1;
return 2*TowerHanoi(N-1) + 1;
}
TowerHanoi:
slti $t1, $a0, 1
bne $t1, $zero, return_zero

ori $t0, $zero, 1
beq $a0, $t0, return_one

addi $a0, $a0, -1
jal TowerHanoi

sll $v0, $v0, 1
addi $v0, $v0, 1
jr $ra

return_zero:
ori $v0, $zero, 0
jr $ra

return_one:
ori $v0, $zero, 1
jr $ra
```

Bài tập 2: Chuyển từ Assembly MIPS sang C

Giả sử f, g, h, i, j = $s0, $s1, $s2, $s3, $s4; A, B = $s6, $s7

a, b, c, d = $a0, $a1, $a2, $a3

```
a) beq $s1, $s2, label
bgt $s2, $s3, exit
ble $s3, $s4, exit
```

```
label: add $s4, $s4, $s1
if (g == h) j = j + g;
else{
    if (g > i) return;
    if (i <= j) return;
    d = d + a;
}

b) li $s3, 10
ori $t0, $zero, 1

do_loop_b:
srl $s3, $s3, 2
bgt $s3, $t0, do_loop_b
i = 10;
do{
    i = i / 4;
} while (i > 1);

c) sll $t0, $s3, 2
add $t0, $t0, $s7
sw $t1, 0($t0)
bne $t1, $s4, else
mul $s2, $s0, $s1
j end_if_c
else:
add $s0, $s1, $s2
end_if_c:
if(B[i] == j) h = f * g;
else f = g + h

d) bge $s3, $s4, else

sll $t0, $s4, 2
add $t0, $t0, $s7
sw $t1, 0($t0)

sub $s0, $t1, $s1
j end_if_d
else:
and $s0, $s1, $s2
end_if_d:
if(i >= j) f = B[j] - g;
else f = g & h;

e) ori $t0, $zero, 5
ori $t1, $zero, 15
```

```
ble $s3, $t0, case_1
bge $s3, $t1, case_2
mul $s0, $s1, $s2
j end_if_z
case_1:
case_2:
add $s0, $s1, $s2
end_if_z:
if (i <= 5 || i >= 15) f = g + h;
else f = g * h;
```