REPORT

보고서 작성 서약서

1. 나는 타학생의 보고서를 복사(Copy)하지 않았습니다.
2. 나는 타학생의 보고서를 인터넷에서 다운로드 하여 대체하지 않았습니다.
3. 나는 타인에게 보고서 제출 전에 보고서를 보여주지 않았습니다.
4. 보고서 제출 기한을 준수하였습니다.

나는 보고서 작성시 위법 행위를 하지 않고,
성.균.인으로서 나의 명예를 지킬 것을 약속합니다.

과 목 : 컴퓨터 구조
과 제 명 : Project phase 2
담 당 교 수 : 윤희용 교수님
학 과 : 컴퓨터공학
학 년 : 3학년
학 번 / 이름 : 2013314967 정기빈
2013311940 이정호
제 출 일 : 2015 / 5 / 28 (목)

***Phase II*: Manually compile the program into MIPS assembly. The report includes the list of the assembly code, and the estimated execution time in terms of instruction count. In the assembly program, put the instruction by which the number of instructions executed can be counted when the program is run.**

- the estimated execution time in terms of instruction count.

IC : 68000

We use count instruction in $v1 register

We use counter

- start of Label
- before/after branch instruction

-MIPS Assembly Code

```
.data
# city Travel[7] struct
Travel: .space 112

#path ppq[1000]
ppq: .space 108000

# int current
current: .word 1

# int now
now: .word 1


        .text                           #begin of code
```

```
        .globl main

main:
                addi $v1, $zero, 40
                addi $sp, $sp, -4
                sw $ra, 0($sp)                  # main return address

                la $t0, Travel                  # $t0 -> Travel[0].x Addr
                l.d $f0, const5($gp)
                s.d $f0, 0($t0)             # Travel[0].x = 5

                s.d $f0, 8($t0)             # Travel[0].y = 5

                l.d $f0, const2($gp)                # Travel[1]
                s.d $f0, 16($t0)

                l.d $f0, const3($gp)
                s.d $f0, 24($t0)

                l.d $f0, const8($gp)                # Travel[2]
                s.d $f0, 32($t0)

                l.d $f0, const4($gp)
                s.d $f0, 40($t0)


                l.d $f0, const7($gp)                # Travel[3]
                s.d $f0, 48($t0)

                l.d $f0, const2($gp)
                s.d $f0, 56($t0)


                l.d $f0, const1($gp)                # Travel[4]
                s.d $f0, 64($t0)

                l.d $f0, const6($gp)
                s.d $f0, 72($t0)
```

```
                l.d $f0, const9($gp)            # Travel[5]
                s.d $f0, 80($t0)

                l.d $f0, const6($gp)
                s.d $f0, 88($t0)



                l.d $f0, const3($gp)            # Travel[6]
                s.d $f0, 96($t0)

                l.d $f0, const2($gp)
                s.d $f0, 104($t0)

                addi $sp, $sp, -108             # assign Sholtest

                l.d $f0, cosnt0($gp)
                s.d $f0, 0($sp)         # Shortest.city[0] = 0

                sw $zero, 72($sp)       # Shortest.current = 0

                sw $zero, 80($sp)       # Shortest.list[0] = 0

                li $t1, 1
                sw $t1, 76($sp)                 # Shortest.size = 1

                l.d $f0, constcosnt0($gp)
                s.d $f0, 64($sp)        # Sortest.distant = 0

                li $s0, 1               # i is $s0, i = 1

for1st:
                addi $v1, $v1, 2
                slti $t5, $s0, 7        # $t0 = 1, if $s0 < 7
                beq $t5, $zero, exit1           # go to exit1 if $s0 >= 7

                addi $v1, $v1, 12
```

```
                sll $t1, $s0, 3                 # $t1 = i * 8

                add $s2, $sp, $t1       # $s2 -> Sholtest.city[i] addr
                li $t3, 100
                sw $t3, 0($s2)                  # Sholtest.city[i] = 100

                sll $t1, $s0, 2                 # $t1 = i * 4
                addi $t2, $sp, 80       # $t2 -> Sholtest.list
                add $s3, $t2, $t1       # $s3 -> Sholtest.list[i] addr
                li $t3, -1
                sw $t3, 0($s3)
                sll $t1, $s0, 4
                add $s4, $t0, $t1       # $s4 -> Travel[i] addr

                li $s1, 1               # $s1 = j

for2st:
                addi $v1, $v1, 4
                slti $t5, $s1, 7        # $t0 = 1, if $s0 < 7
                sll $t1, $s1, 4
                add $s5, $t0, $t1       # Travel[j]

                beq $t5, $zero, exit            # go to exitisv if $s0 >= 7

                addi $v1, $v1, 1
                beq $s0, $s1, noif

                addi $v1, $v1, 6
                move $a0, $s5
                move $a1, $s4
                jal dist                #result -> $f0
                l.d $f2, 0($s2)         # $f2 = Sholtest.city[i]

                c.lt.d $f0, $f2
                bc1f noif2

                addi $v1, $v1, 1
                s.d $f0, 0($s2)
```

```
noif2:
noif:
                addi $v1, $v1, 2
                addu $s1, $s1, 1        # j++
                j for2st

exit2:
                addi $v1, $v1, 2
                addi $s0, $s0, 1
                j for1st

exit1:

                addi $v1, $v1, 5
                move $a0, $sp



                jal ppqpush

                jal Astar


                addi $sp, $sp, 4        # main return address
                jr $ra
```

```
        .globl ppqpush

#parameter path k
# k is $a0 ,
ppqpush:
                addi $v1, $v1, 58
                addi $sp, $sp, -32
                sw $s0, 28($sp)          # reserve $s0 - $s6
                sw $s1, 24($sp)
                sw $s2, 20($sp)
                sw $s3, 16($sp)
                sw $s4, 12($sp)
                sw $s5, 8($sp)
                sw $s6, 4($sp)
                sw $s7, 0($sp)

                la $s2, current          # $s2 -> current address
                lw $s1, 0($t1)           # $s1 = current

                la $t2, now
                lw $s3, 0($t2)           # $s3 = now

                la $s0, ppq              # $s0 -> ppq address

                li $t2, 108              # $t2 = 108 : ppq count
                mul $t2, $s1, $t2 # $t2 = current * 108
                add $t7, $s0, $t2 # $t7 = ppq[current] addr



                l.d $f0, 0($a0)     # $f0 = k.city[0]
                s.d $f0, 0($t7)     # ppq[current].city[0] = $f0

                l.d $f0, 8($a0)     # $f1 = k.city[1] ..
                s.d $f0, 8($t7)     # ppq[current].city[1] = $f2

                l.d $f0, 16($a0)
                s.d $f0, 16($t7)
```

```
l.d $f0, 24($a0)
s.d $f0, 24($t7)

l.d $f0, 32($a0)
s.d $f0, 32($t7)


l.d $f0, 40($a0)
s.d $f0, 40($t7)


l.d $f0, 48($a0)
s.d $f0, 48($t7)


l.d $f0, 56($a0)    # f7 = k.cost
s.d $f0, 56($t7)    # ppq[current].cost = $f1

l.d $f0, 64($a0)    # f8 = k.distant
s.d $f0, 64($t7)    # ppq[current].distant = $f16

lw $t3, 72($a0)             # t4 = k.current
sw $t3, 72($t7)             # ppq[current].current = $t

lw $t3, 76($a0)             # t5 = k. size
sw $t3, 76($t7)

lw $t3, 80($a0)             # $s0 = k.list[0]
sw $t3, 80($t7)             # ppq[current].list[0] = $s0


lw $t3, 84($a0)
sw $t3, 84($t7)

lw $t3, 88($a0)
sw $t3, 88($t7)

lw $t3, 92($a0)
```

```
                sw $t3, 92($t7)

                lw $t3, 96($a0)
                sw $t3, 96($t7)

                lw $t3, 100($a0)
                sw $t3, 100($t7)

                lw $t3, 104($a0)
                sw $t3, 104($t7)


                addi $s1, $s1, 1   # current++
                sw $s1, 0($s2)

                addi $sp, $sp, -108          # assign path temp

                move $s4, $s3                # i = now

for1stpush:
                addi $v1, $v1, 2
                slt $t3, $s4, $s2   # $t4 = 0 if i >= current
                beq $t3, $zero, exitppqpush1

                addi $v1, $v1, 1
                move $s5, $s3                # j = now

for2stpush:
                addi $v1, $v1, 2
                slt $t4, $s5, $s2   # $t5 = 0 if j >= current
                beq $t4, $zero,  exitppqpush2

                addi $v1, $v1, 108
                li $t0, 108
                mul $t1, $t0, $s4 # $t1 = i*108
                mul $t2, $t0, $s5 # $ = j*108

                add $s6, $s0, $t1 # $a1 = ppq[i] addr
                add $s7, $s0, $t2 # $a2 = ppq[j]
```

```
addi $t3, $s6, 56 # $t3 = ppq[i].cost addr
addi $t4, $s7, 56 # $t4 = ppq[j].cost addr

l.d $f0, 0($t3)              # $f0 = ppq[i].cost
l.d $f2, 0($t4)


c.lt.d $f0, $f2              # CC = 0 if ppq[i].cost >=  ppq[j].cost
bc1f noifppqpush             # go to noif.. if  cc =0


                             # temp = ppq[i]

addi $v1, $v1, 2

l.d $f4, 0($s6)      # $f4 = ppq[i].city[0]
s.d $f4, 0($sp)      # temp.city[0] = $f0

l.d $f4, 8($s6)      # $f4 = ppq[i].city[1]
s.d $f4, 8($sp)      # temp.city[1] = $f4

l.d $f4, 16($s6)
s.d $f4, 16($sp)

l.d $f4, 24($s6)
s.d $f4, 24($sp)

l.d $f4, 32($s6)
s.d $f4, 32($sp)


l.d $f4, 40($s6)
s.d $f4, 40($sp)


l.d $f4, 48($s6)
s.d $f4, 48($sp)
```

```
l.d $f4, 56($s6)    # f4 = ppq[i].cost
s.d $f4, 56($sp)    # temp.cost = $f4

l.d $f4, 64($s6)    # f4 = ppq[i].distant
s.d $f4, 64($sp)    # temp.distant = $f4

lw $t3, 72($s6)             # t3 = ppq[i].current
sw $t3, 72($sp)             # temp.current = $t3

lw $t3, 76($s6)             # t3 = ppq[i]. size
sw $t3, 76($sp)

lw $t3, 80($s6)             # $t3 = ppq[i].list[0]
sw $t3, 80($sp)             # temp.list[0] = $s0


lw $t3, 84($s6)
sw $t3, 84($t7)

lw $t3, 88($a0)
sw $t3, 88($t7)

lw $t3, 92($a0)
sw $t3, 92($t7)

lw $t3, 96($a0)
sw $t3, 96($t7)

lw $t3, 100($a0)
sw $t3, 100($t7)

lw $t3, 104($a0)
sw $t3, 104($t7)


                            # ppq[i] = ppq[j]

l.d $f4, 0($s7)
```

```
s.d $f4, 0($s6)

l.d $f4, 8($s7)
s.d $f4, 8($s6)

l.d $f4, 16($s7)
s.d $f4, 16($s6)

l.d $f4, 24($s7)
s.d $f4, 24($s6)

l.d $f4, 32($s7)
s.d $f4, 32($s6)


l.d $f4, 40($s7)
s.d $f4, 40($s6)


l.d $f4, 48($s7)
s.d $f4, 48($s6)


l.d $f4, 56($s7)
s.d $f4, 56($s6)

l.d $f4, 64($s7)
s.d $f4, 64($s6)

lw $t3, 72($s7)
sw $t3, 72($s6)

lw $t3, 76($s7)
sw $t3, 76($s6)

lw $t3, 80($s7)
sw $t3, 80($s6)
```

```
lw $t3, 84($s7)
sw $t3, 84($s6)

lw $t3, 88($s7)
sw $t3, 88($s6)

lw $t3, 92($s7)
sw $t3, 92($s6)

lw $t3, 96($s7)
sw $t3, 96($s6)

lw $t3, 100($s7)
sw $t3, 100($s6)

lw $t3, 104($s7)
sw $t3, 104($s6)
```

ppq[j] = temp

```
l.d $f4, 0($sp)
s.d $f4, 0($s7)

l.d $f4, 8($sp)
s.d $f4, 8($s7)

l.d $f4, 16($sp)
s.d $f4, 16($s7)

l.d $f4, 24($sp)
s.d $f4, 24($s7)

l.d $f4, 32($sp)
s.d $f4, 32($s7)


l.d $f4, 40($s7)
s.d $f4, 40($s6)
```

```
l.d $f4, 48($sp)
s.d $f4, 48($s7)


l.d $f4, 56($sp)
s.d $f4, 56($s7)

l.d $f4, 64($sp)
s.d $f4, 64($s7)

lw $t3, 72($sp)
sw $t3, 72($s7)

lw $t3, 76($sp)
sw $t3, 76($s7)

lw $t3, 80($sp)
sw $t3, 80($s7)


lw $t3, 84($sp)
sw $t3, 84($s7)

lw $t3, 88($sp)
sw $t3, 88($s7)

lw $t3, 92($sp)
sw $t3, 92($s7)

lw $t3, 96($sp)
sw $t3, 96($s7)

lw $t3, 100($sp)
sw $t3, 100($s7)

lw $t3, 104($sp)
sw $t3, 104($s7)
```

```
noifppqpush:
                addi $v1, $v1, 2
                addi $s5, $s5, 1   # j++
                j for2stpush
 exitppqpush2:
                addi $v1, $v1, 2
                addi $s4, $s4, 1   # i ++
                j for1stpush

 exitppqpush1:
                addi $v1, $v1, 10
                lw $s7, 0($sp)
                lw $s6, 4($sp)              # restore $s0 - $s7
                lw $s5, 8($sp)
                lw $s4, 12($sp)
                lw $s3, 16($sp)
                lw $s2, 20($sp)
                lw $s1, 24($sp)
                lw $s0, 28($sp)
                addi $sp, $sp, 32
                jr $ra




        .globl checkgoal

# check goal parameter ; path P
# result is $v0, P is $a0
checkgoal:
                addi $v1, $v1, 8
                addi $sp, $sp, -4
                sw $s0, 0($sp)


                li $t0, 24          # $t0 = 24
                addi $t1, $a0, 80 # $t1 -> P.list addr
                addu $t1, $t1, $t0          # $t1 -> P.list[6] addr
```

```
                lw $s0, 0($t1)
                li $t2, 6               # $t2 = 6

                bne $s0, $t2, false             # if(p.list[6] ==6)

                addi $v1, $v1, 4
                li $v0, 1

                lw $s0, 0($sp)
                addi $sp, $sp, 4
                jr $ra

false:
                addi $v1, $v1, 4
                move $v0, $zero
                lw $s0, 0($sp)
                addi $sp, $sp, 4
                jr $ra

        .globl dist

# distparameter ; city a, city b
# result is $f0, a is in $a0, b is $a1

dist:
        addi $v1, $v1, 16
        addi $sp, $sp, -4
        sw $ra, 0($sp)

        l.d $f2, 0($a0)
        l.d $f4, 4($a0)
        l.d $f6, 8($a0)
        l.d $f8, 12($a0)

        sub.d $f10, $f2, $f6
        sub.d $f12, $f4, $f8
        mul.d $f10, $f10, $f10
        mul.d $f12, $f12, $f12
        add.d $f30, $f12, $f10
```

```
        l.d $f28, const1($gp)
        jal sqrt

        lw $ra, 0($sp)
        addi $sp, $sp, 4
        jr $ra


        .globl sqrt

# sqrtparameter ; double a.x, double b.x
# result is $f0, input is in $f30, x is in $f28
sqrt:
                addi $v1, $v1, 4
                addi $sp, $sp, -4
                sw $s0, 0($sp)

                move $s0, $zero         # i = 0
forsqrt:
                addi $v1, $v1, 2
                slti $t0, $s0, 10       # $t0 = 0 if i >= 10
                beq $t0, $zero, exitsqrt

                addi $v1, $v1, 6
                div.d $f2, $f30, $f28           # $f2 = input / x
                add.d $f2, $f2, $f28            # $f2 = x + (input / x)
                l.d $f8, const2($gp)            # $f8 = 2
                div.d $f0, $f2, $f8     # $f0 = x + (input / x) / 2

                addi $s0, $s0, 1        # i = i+1
                j forsqrt
exitsqrt:
                addi $v1, $v1, 1
                jr $ra

        .globl PUSH

# PUSHparameter ; path now, int ncity
# now is $a0, ncity is $a1
```

PUSH:

```
addi $v1, $v1, 102
addi $sp, $sp, -32
sw $s0, 28($sp)
sw $s1, 24($sp)
sw $s2, 20($sp)
sw $s3, 16($sp)
sw $s4, 12($sp)
sw $s5, 8($sp)
sw $s6, 4($sp)
sw $ra, 0($sp)

addi $sp, $sp, -108          # assign next
l.d $f0, 0($a0)      # $f0 = now.city[0]
l.d $f2, 8($a0)      # $f1 = now.city[1] ..
l.d $f4, 16($a0)
l.d $f6, 24($a0)
l.d $f8, 32($a0)
l.d $f10, 40($a0)
l.d $f12, 48($a0)

l.d $f14, 56($a0)  # f7 = now.cost
l.d $f16, 64($a0)  # f8 = now.distant
lw $t4, 72($a0)              # t4 = now.current
lw $t5, 76($a0)              # t5 = now. size

lw $s0, 80($a0)              # $s0 =now.list[0]
lw $s1, 84($a0)
lw $s2, 88($a0)
lw $s3, 92($a0)
lw $s4, 96($a0)
lw $s5, 100($a0)
lw $s6, 104($a0)


s.d $f0, 0($sp)      # next.city[0] = $f0
s.d $f2, 8($sp)      # next.city[1] = $f2
s.d $f4, 16($sp)
s.d $f6, 24($sp)
```

```
s.d $f8, 32($sp)
s.d $f10, 40($sp)
s.d $f12, 48($sp)

s.d $f14, 56($sp)  # next.cost = $f14
s.d $f16, 64($sp)  # next.distant = $f16
sw $t4, 72($sp)            # ppq[current].current = $t
sw $t5, 76($sp)            # t5 = k. size

sw $s0, 80($sp)            # $s0 = k.list[0]
sw $s1, 84($sp)
sw $s2, 88($sp)
sw $s3, 92($sp)
sw $s4, 96($sp)
sw $s5, 100($sp)
sw $s6, 104($sp)

addi $sp, $sp, -8
sw $a0, 0($sp)
sw $a1, 4($sp)

                                # call dist
la $t0, Travel                  # $t0 -> Travel Address
addi $t1, $a0, 72          # $t1 -> now.current addr

lw $t2, 0($t1)                  # $t2 = now.current
sll $t2, $t2, 4                 # $t2 = now.current * 16
add $a0, $t0, $t2          # $a0 -> Travel[now.current] Address

sll $t3,$a1, 4                  # $a1 = ncity * 16
add $a1, $t0, $t3          # $a1 -> Trave[ncity] Address

jal dist


lw $a0, 0($sp)
lw $a1, 4($sp)                  # $a1 = ncity

addi $sp, $sp, 8           # restore $sp
```

```
sll $t4, $a1, 3                 # $t4 = ncity * 8
add $t5, $sp, $t4           # $t5 -> next.city[ncity] Address
s.d $f0, 0($t5)

addi $t4, $sp, 72           # $t4 -> next.current Address
sw $a1, 0($t4)                  # next.current = ncity

l.d $f0, 0($sp)             # $f0 = next.city[0]
l.d $f2, 8($sp)             # $f2 = next.city[0]
add.d $f0, $f0, $f2             # $f0 = next.city[0]+ next.city[1]

l.d $f2, 16($sp)            # $f2 = next.city[2]
add.d $f0, $f0, $f2             # $f0 = $f0 + next.city[2]


l.d $f2, 24($sp)            # $f2 = next.city[3]
add.d $f0, $f0, $f2             # $f0 = $f0 + next.city[3]
l.d $f2, 32($sp)            # $f2 = next.city[4]
add.d $f0, $f0, $f2             # $f0 = $f0 + next.city[4]
l.d $f2, 40($sp)            # $f2 = next.city[5]
add.d $f0, $f0, $f2             # $f0 = $f0 + next.city[5]
l.d $f2, 48($sp)            # $f2 = next.city[6]
add.d $f0, $f0, $f2             # $f0 = $f0 + next.city[6]

addi $t1, $sp, 56           # $t1 -> next. cost Address
s.d $f0, 0($t1)             # next.cost = $f0

add $t2, $a0, 76            # $t2 -> now.size Address
lw $t3, 0($t2)                  # $t3 = now.size
sll $t5, $t3, 2                 # $t5 = now.size * 4
addi $t4, $sp, 80           # $t4 -> next.list Address
add $t4, $t4, $t5           # $t4 -> next.list[now.size]
sw $a1, 0($t4)                  # next.list[now.size] = ncity

addi $t3, $t3, 1            # $t3 = now.size + 1
sw $t3, 76($sp)                 # next.size = $t3

move $a0, $sp                   # call ppqpush
```

```
                la $a3,ppq

                jal ppqpush


                addi $sp, $sp, 108              # restore $sp

                lw $ra, 0($sp)
                lw $s6, 4($sp)
                lw $s5, 8($sp)
                lw $s4, 12($sp)
                lw $s3, 16($sp)
                lw $s2, 20($sp)
                lw $s1, 24($sp)
                lw $s0, 28($sp)
                addi $sp, $sp, 32

                jr $ra

        .globl proceduremove

# proceduremove parameter : path p
# p is in $a0; global: we use current is in $a1, now is in in $a2

proceduremove:
                addi $v1, $v1, 6
                addi $sp, $sp, -8       # reserve $s0, $ra
                sw $s0, 4($sp)
                sw $ra, 0($sp)

                lw $t0, 0($a1)                  # $t0 = current
                lw $t1, 0($a2)                  # $t1 = now
                beq $t0, $t1, noifmove          # go to noifmove if current == now

                addi $v1, $v1, 62
                addi $t1, $t1, 1        # $t1 = now +1
                sw $t1, 0($a2)                  # now++
```

```
noifmove:
                addi $v1, $v1, 61
                move $s0, $zero                 # i is $s0
    formove:
                addi $v1, $v1, 2
                slti $t0, $s0, 7            # $t0 = 1, if $s0 < 7
                beq $t0, $zero, exitmove        # go to exitmove if $s0 >= 7

                addi $v1, $v1, 4
                add $t2, $a0, 76            # $t2 -> p.size Address
                lw $t3, 0($t2)                  # $t3 = p.size
                li $t4, 7

                beq $t3, $t4, noif1stmove # go to noif2stmove if p.size == 7

                addi $v1, $v1, 3
                move $a1, $s0                   # call isvisited
                jal isvisited
                bne $v0, $zero, noif2stmove     # go to noif2stmove if isvisited(p, i) != false

                addi $v1, $v1, 1
                jal PUSH                    # call

 noif2stmove:

 noif1stmove:

    exitmove:
                addi $v1, $v1, 4
                lw $ra, 0($sp)
                lw $s0, 4($sp)
                addi $sp, $sp, 8

                jr $ra

        .globl isvisited

# isvisited; parameter : path p, int a; returntype : boolean
# p is $a0, a is $a1, result is $v0
```

```
isvisited:
        addi $v1, $v1, 6
        addi $sp, $sp, -12
        sw $s2, 8($sp)              # reserve $s0, $s1, $s2
        sw $s1, 4($sp)
        sw $s0, 0($sp)

        move $v0, $zero             #return false
        move $s0, $zero             # s0 is i, i = 0
 forisv:
        addi $v1, $v1, 2
        slti $t0, $s0, 7        # $t0 = 1, if $s0 < 7
        beq $t0, $zero, exitisv         # go to exitisv if $s0 >= 7

        addi $v1, $v1, 5
        sll $t1, $s0, 2             # $t1 = i * 4
        addi $s2, $a0, 80       # $s2 -> k.list Address
        add $t2, $t1, $s2       # $t2 -> k.list[i] Address
        lw $t3, 0($t2)              # $t3 = k.list[i]

        bne $t3, $a1, noifisv           # go to noifisv if $t3 != $a1

        addi $v1, $v1, 2
        li $v0, 1               # return true
        j exitisv

 noifisv:
        addi $v1, $v1, 2
        addi $s0, $s0, 1        # i++
        j forisv

 exitisv:
        addi $v1, $v1, 5
        lw $s0, 0($sp)
        lw $s1, 4($sp)
        lw $s2, 8($sp)              #restore $s0, $s1, $s2
        addi $sp, $sp, 12
        jr $ra
```

```
        .globl Astar

# Astar parameter path p
# p is $a1

  Astar:
                addi $v1, $v1, 5
                addi $sp, $sp, -12
                sw $s0, 8($sp)                  # end is $s0
                sw $s1, 4($sp)                  # i is $s1
                sw $ra, 0($sp)

                move $s1, $zero                 # i = 0
forAstar:
                addi $v1, $v1, 2
                slti $t0, $s0, 7          # $t0 = 1, if $s0 < 7
                beq $t0, $zero, exitAstar  # go to exitAstar if $s0 >= 7

                addi $v1, $v1, 10
                addi $t1, $a1, 80         # $t1 -> p.list Address
                sll $t2, $s1, 2                 # $t2 = i * 4
                add $t1, $t1, $t2         # $t1 -> p.list[i]
                lw $a0, 0($t1)                  # $a0 = p.list[i]
                addi $a0, $a0, 1
                li $v0, 1
                syscall

                move $a0, $a1                   # call checkgoal
                jal checkgoal                   # result is $v0

                beq $v0, $zero, noifAstar  # go to noifAstar if $v0 == 0

                addi $v1, $v1, 2
                li $v0, 10
                syscall

noifAstar:
```

```
                addi $v1, $v1, 7
                jal proceduremove

                la $a1, ppq
                lw $t3, 0($a2)                  # $t3 = now
                li $t5, 108
                mul $t4, $t3, $t5       # $t4 = now * 108

                add $a1, $a1, $t4
                jal Astar

exitAstar:
                addi $v1, $v1, 5
                lw $ra, 0($sp)
                lw $s1, 4($sp)
                lw $sp, 8($sp)
                addi $sp, $sp, 12
                jr $ra
```