COSC 2406 TEST 19 Mar, 2018

NO SILICON DEVICES ARE ALLOWED.

1. Do all calculations in this question in base 16. [ 8]

a) Calculate the 4 hex digit 2C representation of the integer -11826.

16 | 11286

-----

16 | 739 - 2

---

16 | 46 - 3

--

16 | 2 - 14

-

0 - 2

So 11826 <-> 2E32h

1C <-> D1CD

2C <-> DICE

-11826 <-> DICE

b) Perform the following unsigned subtraction.

A7B7h - 5BC9h

A7B7

- 5BC9

----

4BEE

c) Perform the following signed subtraction using 2C addition.

A5B6h - B7FCh

A5B6 --+

- B7FC |

---- |

1C 4803 +

2C 4804 |

---- |

EDBA --+

d) Indicate whether there is signed and/or unsigned overflow in the

above example (justify your answer).

unsigned

ovf since minuend < subtrahend

signed

neg - neg <-> neg + pos <-> pos - pos

Shown in class that difference of two integers can never lead

to overflow

2. a) The 8086 ISA has 8 general purpose registers. Name them. [ 8]

AX, BX, CX, DX, BI, SI, BP, SP

b) Name the memory segments in the 8086 ISA and indicate the register

that points to each segment.

stack seg: SS

data seg: DS

code seg: CS

extra seg: ES

c) Describe how an assembler assigns addresses to variables and

labels in an AL program.

LC <- 0 at beginning of data/code seg

LC <- LC + sizeof data/instruction

addr of variable/ label <- LC

d) Explain the benefit derived from separating the 8086 chip into

an execution unit and a bus interface unit.

increased instruction throughput, speedup

pipelining - execiting instruction in parallel with next

instruction fetch

3. Consider the following expression. [20]

x/(a + b) + (x\*x - a\*a)\*(a\*x + b\*y - 5)

a) Draw the AST for the above expression.

3 + AX

|

+-----------------------------+

| |

2 div 3 \* AX/BX

/ \ |

/ \ +--------------------+

AX x 1 + CX | |

1 / \ | |

a b 2 - AX 2 - BX

1 0 / \ / \

CX / \ / \

1 \* AX 1 \* AX/CX 2 + BX 5

/ \ / \ / \ 0

x x a a / \

1 0 1 0 / \

/ \

1 \* AX/BX 1 \* AX

/ \ / \

a x b y

1 0 1 0

b) Label the AST with Sethi-Ullman numbers in order to calculate the

minimum number of registers required to evaluate the expression.

c) Use the Sethi-Ullman algorithm to write optimal 8086 AL code to

evaluate the above expression.

starting with busiest tree

mov ax, a

imul x

mov bx, ax

mov ax, b

imul y

add bx, ax

sub bx, 5

mov ax, a

imul ax

mov cx, ax

mov ax, x

imul ax

sub ax, cx

imul bx

mov bx, ax

mov cx, a

add cx, b

mov ax, x

cwd

idiv cx

add ax, bx

4. Translate the following statements into 8086 AL code. Assume that [20]

i ) all variables have been allocated memory space, and

ii) k >= 0.

Use the LOOP AL instruction for each FOR statement.

s := 0;

for y := 0 to k do

for x := 0 to k do

if x\*y <= k then

s := s + 1

endif

endfor

endfor;

; s := 0

mov s, 0

; for01 y := 0 to k

mov cx, k

inc cx

mov y, 0

do01:

; for02 x := 0 to k

push cx

mov cx, k

inc cx

mov x, 0

do02:

; if03 x\*y <= k

mov ax, x

imul y

cmp ax, k

jnle endif03

; then03

inc s

endif03:

inc x

loop do02

; endfor02

pop cx

inc y

loop do01

; endfor01

5. Translate the procedure PellNums into 8086 AL code in a complete [34]

module. Use GetInt and PutInt for integer I/O.

procedure PellNums();

var

i, n, p1, p2, pn: integer;

begin

loop

write 'n? ';

read n;

exit when n < 0;

if n = 0 then

pn := 0

elsif n = 1 then

pn := 1

elsif n = 2 then

pn := 2

else

i := 2;

p1 := 1;

p2 := 2;

repeat

pn := 2\*p2 + p1;

p1 := p2;

p2 := pn;

i := i + 1

until i = n

endif;

write pn, cr, lf

endloop

end PellNums;

title pellnums.asm

.model small

.stack 100h

.data

include const.in

i dw ?

n dw ?

p1 dw ?

p2 dw ?

pn dw ?

nq db 'n? $'

crlf db cr, lf, '$'

.code

extrn getint: proc, putint: proc

main proc

mov ax, @data

mov ds, ax

loop01:

; write 'n? '

mov ah, wrstr

mov dx, offset nq

int dosfunc

; read n

call getint

mov n, ax

; exit when n < 0

cmp n, 0

jl endloop01

; if02 n = 0

cmp n, 0

jne elsif01a

; then02

; pn := 0

mov pn, 0

jmp endif02

elsif02a: ; n = 1

cmp n, 1

jne elsif02b

; then02a

; pn := 1

mov pn, 1

jmp endif02

elsif02b: ; n = 2

cmp n, 2

jne else02

; then02b

; pn := 2

mov pn, 2

jmp endif02

else02:

; i := 2

mov i, 2

; p1 := 1

mov p1, 1

; p2 := 2

mov p2, 2

repeat02:

; pn := 2\*p2 + p1

mov ax, 2

imul p2

add ax, p1

mov pn, ax

; p1 := p2

mov ax, p2

mov p1, ax

; p2 := pn

mov ax, pn

mov p2, ax

; i := i + 1

inc i

; until03 i = n

mov ax, i

cmp ax, n

jne repeat02

; endrepeat02

endif02:

; write pn, cr, lf

mov ax, pn

call putint

mov ah, wrstr

mov dx, offset crlf

int dosfunc

jmp loop01

endloop01:

mov ah, ret2dos

int dosfunc

main endp

end main

total [90]