System Programing

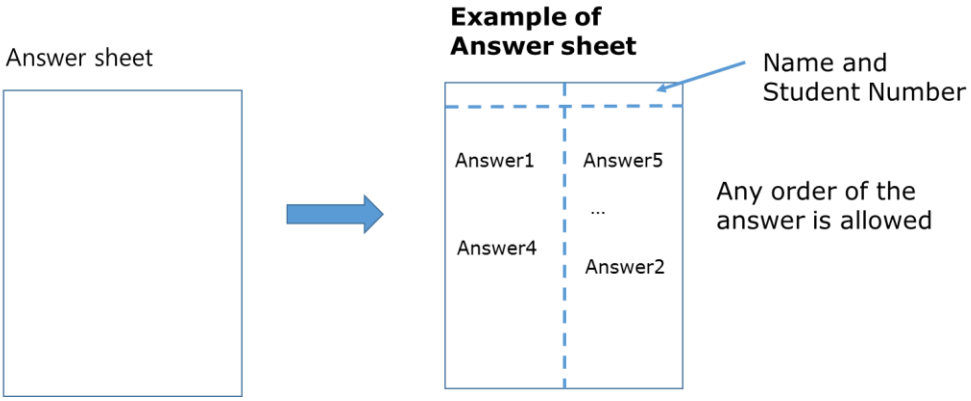
Midterm Examination, Fall 2018

Pusan National University

October 24, 2018

Time Limit: 75 minutes (13:30 – 14:45)

- Please write your answers on the answer sheet. Halve down your answer sheet and give top margins for your information. Follow the instructions shown in the following figure. If you do not follow the example, you will loose 1o% points from your total score. **Do not use the back side of the answer sheet.**



- Don't forget to write your name and student number (StudentID).
- Closed notes; closed book; no sheet of formulas permitted.
- In case you find a question ambiguous, please write down your assumption and answer the question accordingly. Don't ask any questions to T.As.

Turn over the page when instructed to do so. Good Luck!

참고용 표 (문제마다 적절하게 이용할 것)

Summary of Section 4.1.2-4.1.3: Y86 instruction set

Byte	0	1	2	3	4	5
halt	0	0				
nop	1	0				
rrmovl rA, rB	2	0	rA	rB		
irmovl V, rB	3	0	F	rB	V	
rmmovl rA, D(rB)	4	0	rA	rB	D	
mrmovl D(rB), rA	5	0	rA	rB	D	
OP1 rA, rB	6	fn	rA	rB		
jXX Dest	7	fn	Dest			
cmovXX rA, rB	2	fn	rA	rB		
call Dest	8	0	Dest			
ret	9	0				
pushl rA	A	0	rA	F		
popl rA	B	0	rA	F		

Number	Register Name
0	%eax
1	%ecx
2	%edx
3	%ebx
4	%esp
5	%ebp
6	%esi
7	%edi
F	No register

Program register identifiers

fn	jump
0	jmp
1	jle
2	jl
3	je
4	jne
5	jge
6	jg

7 jump functions

fn	operation
0	addl
1	subl
2	andl
3	xorl

Operations supported

Operations

addl	6 0
subl	6 1
andl	6 2
xorl	6 3

Branches

jmp	7 0	jne	7 4
jle	7 1	jge	7 5
jl	7 2	jg	7 6
je	7 3		

Moves

rrmovl	2 0	cmovne	2 4
cmovle	2 1	cmovge	2 5
cmovl	2 2	cmovg	2 6
cmove	2 3		

Powers of 2:

$2^0 = 1$	
$2^1 = 2$	$2^{-1} = 0.5$
$2^2 = 4$	$2^{-2} = 0.25$
$2^3 = 8$	$2^{-3} = 0.125$
$2^4 = 16$	$2^{-4} = 0.0625$
$2^5 = 32$	$2^{-5} = 0.03125$
$2^6 = 64$	$2^{-6} = 0.015625$
$2^7 = 128$	$2^{-7} = 0.0078125$
$2^8 = 256$	$2^{-8} = 0.00390625$
$2^9 = 512$	$2^{-9} = 0.001953125$
$2^{10} = 1024$	$2^{-10} = 0.0009765625$

Hex help:

```
0x00 = 0
0x0A = 10
0x0F = 15
0x20 = 32
0x28 = 40
0x2A = 42
0x2F = 47
```

1. Number Representation (12 points)

(1) 2's complement 의 2 가지 장점에 대하여 기술하시오. 각 장점당 1 줄 로. 각 장점당 2 줄 이상 기술하면 -2 점 감점. (3 점)

(2) Add **11011001** and **01100011** as two's complement 8 bit integers & convert the result to decimal notation. 올바른 중간 과정을 기술하면 부분 점수를 받을 수 있음. (3 점)

(3) 110101_2 를 unsinged 6-bit int 로 해석할 때, 10 진수로 얼마인가? (3 점)

(4) 110101_2 를 singed (2's complement) 6-bit int 로 해석할 때, 10 진수로 얼마인가? (3 점)

2. Floating-point number representation (20 points)

다음과 같은 표기법을 따르는 floating point 표현 방법을 고려하여 아래의 문제를 푸시오.

Sign (1)	Exponent (4)	Mantissa (3)

(1) 이 새 표현법에서 Exponent 의 bias 값은 얼마인가? (5 points)

(2) 0b 1100 1100 을 10 진수로 변환하라. (5 points)

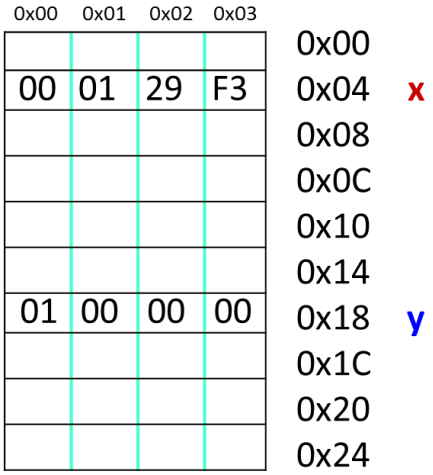
다음과 같은 표기법을 따르는 16 비트 floating point 표현 방법을 고려하여 아래의 문제를 푸시오.

Sign (1)	Exponent (6)	Mantissa (9)

(3) 10 진수 3.625 를 위의 16 비트 floating point 표기법으로 변환하라. 답을 적을 때 1 자리 , 7 자리 다음에 공간을 두어 읽기 쉽게 적으시오(예 0 000111 100001111). 답은 틀려도 계산 과정이 맞을 경우 부분 점수 있음. (10 points)

3. Memory (15 점)

아래 그림은 32-bit 머신에서 메모리의 일부분을 보여 주고 있다. C 언에서 int x, y; 문장을 수행하고 나며 x 는 Ox04 번지, y 는 Ox18 번지로 지정되었다. 다음 각 문장을 수행할 때 마다 메모리의 값은 어떻게 저장되는지 그 값을 적으시오. 이 때 little endian 형식의 머신이라고 가정한다.



(1) x=0; Ox04 번지의 값은? (3 점)

(2) y=Ox3CD02700; Ox18 번지의 값은? (3 점)

(3) x=y+3; Ox04 번지의 값은? (3 점)

(4) int *z = &y + 3; z 가 Ox20 번지라고 가정. Ox20 번지의 값은? (3 점)

(5) *z=y; 몇 번지에 값이 저장되는가? 그리고 그 값은? (3 점)

4. Y86 Assembly encoding (15 점)

다음 Y86 프로그램을 Byte encoding 으로 변환하라. ".pos 0x100"은 개체 코드의 시작 주소가 0x100 이어야 함을 의미한다. 뒤의 문서를 참고한다.

```
.pos   0x100                    # start code at address 0x100
       irmovl  $15, %ebx        # load 15 into %ebx
       rrmovl  %ebx, %ecx       # copy 15 to %ecx
loop:                           # loop
       rmmovl %ecx, -3(%ebx)    # save %ecx at address 15-3=12
       addl       %ebx, %ecx    # increment %ecx by 15
       jmp   loop               # Goto loop
```

```
0x100:  (1)
0x106:  (2)
0x108:  (3)
0x10e:  (4)
0x110:  (5)
```

5. Y86 Assembly decoding (26 점)

다음 나열된 각 바이트 시퀀스에 대해 그것들이 인코딩한 Y86 이 명령어 시퀀스를 작성하라. 만약 시퀀스에 유효하지 않은 바이트가 있으면 해당 위치까지 명령 순서를 표시하고 잘못된 값이 발생한 위치를 표시한다. (1)(2)번 각 4 점, 나머지 3 점

```
0x100: 30f3fcffffff40630008000000
 0x100:  (1)
 0x106:  (2)
 0x10c:  (3)
```

```
0x200: a06f80080200000030f30a00000090
0x200:  (4)
0x202:  (5)
0x207:  (6)
0x208:  proc
0x208:  (7)
0x20e:  (8)
```

6. Assembler code (12 점)

다음은 아주 간단한 어셈블러 코드 exit.s 이다.

```
.section .data
.section .text
.global _start

_start:
movl $1, %eax

movl $0, %ebx

int $0x80
```

(1) 이 소스 프로그램을 어셈블 하는 명령어를 적으시오. (exit.o 생성) (3 점)

(2) 이 소스 프로그램 exit 실행 파일로 링크하는 명령어를 적으시오. (3 점)

(3) .section .data 의 의미를 간단하게 1 줄로 설명하시오. (3 점)

(4) Int $0x80 의 동작을 간단하게 1 줄로 설명하시오. (3 점)