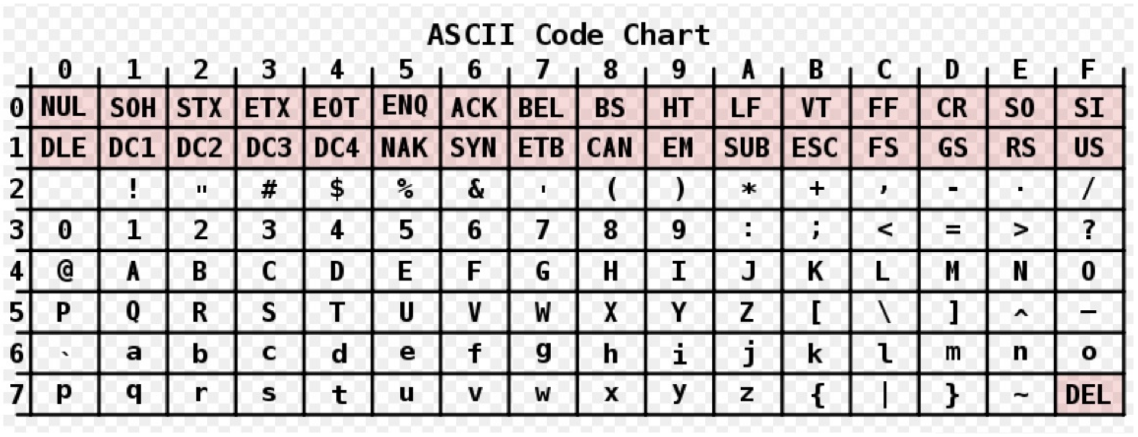
Question 1 (20 points): In one of your CMPUT 229 assignments you need to print the hexadecimal representation of the 32-bit number stored in a given register. You have already created a subroutine called `PrintChar` that prints the character represented in ASCII by the 8 least significant bits of `$a0` to the screen. Now you have to write the subroutine that receives as a parameter the 32-bit number whose hexadecimal representation you want to print and prints one hexadecimal digit at a time. Lets call this subroutine `PrintHex`. To print the alpha characters of the ASCII code, `PrintHex` uses capital letters. `PrintHex` receives in `$a0` the 32-bit number to be printed, and calls `PrintChar` to print each individual hexadecimal digit. The figure below has the ASCII code. For instance, the ASCII code for the character `R` is `0x52`. `PrintHex` must follow all the calling conventions and restrictions on operand sizes of the MIPS architecture. (Please use the next page to write your code and write it clearly).



ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

```
# Register usage:
#       $s0: mask
#       $s1: shfamount
#       $t2: temp
#       $t3: digit
#       $s5: number
PrintHex:
          sub  $sp, $sp,16
          sw   $ra, 0($sp)     # save $ra
          sw   $s0, 4($sp)     # save $s0
          sw   $s1, 8($sp)     # save $s1
          sw   $s5, 12(sp)     $ save $s5
          add  $s5, $a0, $zero # number <- $a0
          lui  $s0, 0xF000     # mask <- 0xF000 0000
          li   $s1, 28         # shfamount <- 28
nex_digit:
          beq  $s0, $zero, done
          and  $t2, $s5, $s0   # temp <- number & mask
          srlv $t3, $t2, $s1   # digit <- temp >> shfamount
          bgt  $t3, 9, alpha   # if digit > 9 then it is alpha
          add  $a0, $t3, 0x30  # char <- digit + 0x30
          b    print
alpha:
          add  $a0, $t3, 0x41  # char <- digit + 0x41
print:
          jal  PrintChar
          srl  $s0, $s0, 4     # mask <- mask >> 4
          subi $s1, $s1, 4     # shfamount <- shfamount - 4
          b    next_digit
done:
          lw   $ra, 0($sp)     # restore $ra
          lw   $s0, 4($sp)     # restore $s0
          lw   $s1, 8($sp)     # restore $s1
          lw   $s5, 12(sp)     $ restore $s5
          add  $sp, $sp, 16
```