NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

MID-TERM TEST
AY2016/17 Semester 2

CS2100 — COMPUTER ORGANISATION

8 March 2017 Time Allowed: **1 hour 30 minutes**

INSTRUCTIONS

1. This question paper contains **TEN (10)** questions (excluding the bonus question) and comprises **NINE (9)** printed pages.
2. An **Answer Sheet**, comprising **TWO (2)** printed page, is provided for you.
3. Write your **Name**, **Matriculation Number** and **Tutorial Group Number** on the Answer Sheet with a **PEN**.
4. Answer **ALL** questions within the space provided on the Answer Sheet.
5. You may write your answers in pencil (at least 2B).
6. Submit only the Answer Sheet at the end of the test. You may keep the question paper.
7. This is a **CLOSED BOOK** test. However, an A4 single-sheet double-sided handwritten reference sheet is allowed.
8. Maximum score is **40 marks**.
9. Calculators and computing devices such as laptops and PDAs are not allowed.
10. Pages 7 and 8 are for your rough work. They contain blank truth tables, K-maps and state table for your use.
11. Page 9 contains the **MIPS Reference Data** sheet.

——— END OF INSTRUCTIONS ———

Bonus question:

0. [This is the bonus question which is worth 1 mark. The mark of this question will only be added if the total mark scored is less than 40.]

   On 2nd February 2017, Aaron shared a story in his lecture about cookies. What is the title of that story?

   A. "Four-and-three-quarter cookies"
   B. "Six-and-a-half cookies"
   C. "Seven cookies"
   D. "How to cook cookies like a booky rooky"
   E. "I have a cookie… I have a monster… ah, cookie monster!"
   F. I think this must be a trick question. He didn't tell any story on that day, but he nearly fainted as he didn't take any cookies on that day.

Questions 1 – 5: Each multiple-choice-question has only <u>one</u> correct answer. Write your answers in the boxes on the **Answer Sheet**. Two marks are awarded for each correct answer and no penalty for wrong answer.

1. Given the following hexadecimal representation in IEEE 754 single-precision floating-point number system:

   C 0 B C 0 0 0 0

   What decimal value does it represent?

   A. -1.875
   B. -5.875
   C. -15×2^{124}
   D. -47×2^{124}
   E. None of the above.

2. Given the following MIPS code fragment, encode the **bne** instruction. (The MIPS Reference Data sheet is provided on page 9.)

```
here: addi $t1, $t1, 2
      add  $t2, $t2, $t1
      bne  $t2, $t0, here
```

   A. 150AFFFE
   B. 1548FFFE
   C. 1548FFFD
   D. 150AFFFD
   E. None of the above.

3. What is the following Boolean expression in ΠM form?

$$F3(W,X,Y,Z) = Y' \cdot W + Y' \cdot Z + Z' \cdot Y \cdot X$$

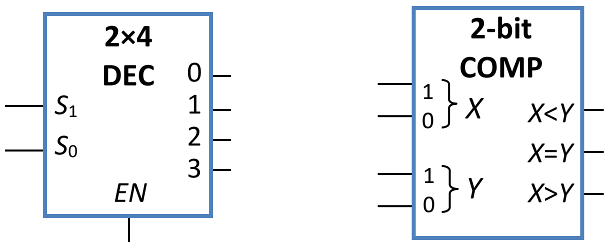
   A. $F3(W,X,Y,Z) = \Pi M(0, 2, 3, 4, 7, 10, 11, 15)$
   B. $F3(W,X,Y,Z) = \Pi M(1, 5, 6, 8, 9, 12, 13, 14)$
   C. $F3(W,X,Y,Z) = \Pi M(0, 2, 8, 9, 10, 12, 13, 14)$
   D. $F3(W,X,Y,Z) = \Pi M(0, 2, 3, 4, 5, 10, 11, 14)$
   E. None of the above.

4. Simplify the following expression of a 15-variable Boolean function. *m*'s are the minterms and *M*'s the maxterms.

$$(m3125 \cdot M987 \cdot m1025) + m895 \cdot (M2222 + M618)$$

   A. 0
   B. 1
   C. *M*987
   D. *m*895
   E. *M*987 + *m*895

5. You are given a 2×4 decoder with 1-enable and active high outputs, and a 2-bit magnitude comparator, as shown below.



Which of the following Boolean functions can be implemented using either or both of the above devices <u>without</u> any additional logic gate? Note that complemented literals are <u>not</u> available. Logical constants 0 and 1 are always available.

   (i) $F5a(A, B, C, D) = A \cdot B \cdot C'$
   (ii) $F5b(A, B, C, D) = \Sigma m(1,2,3)$
   (iii) $F5c(A, B, C, D) = \Sigma m(1,3,5,7)$

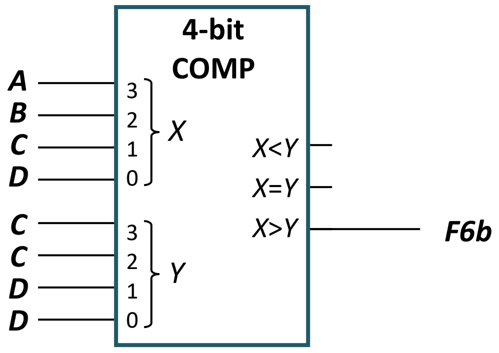
   A. Only (i).
   B. Only (i) and (ii).
   C. Only (i) and (iii).
   D. Only (ii) and (iii).
   E. All of (i), (ii) and (iii).

Questions 6 – 10: Write your answer in the space provided on the **Answer Sheet**. You do not need to show workings, unless otherwise stated.

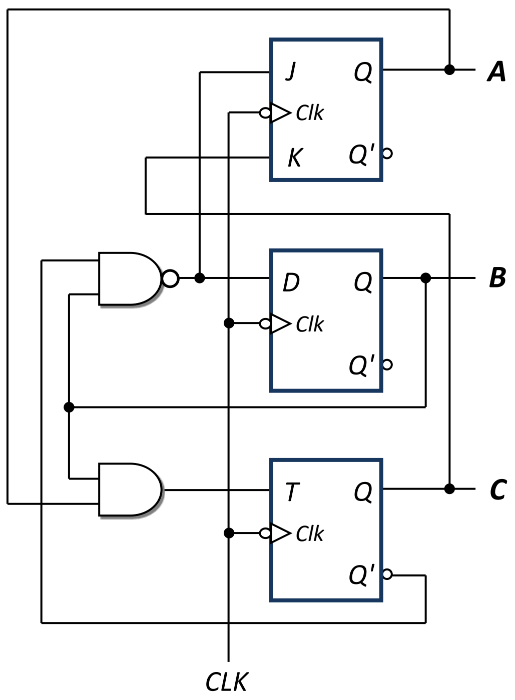
6. [5 marks]
   (a) How many PIs (prime implicants) and EPIs (essential prime implicants) are there in the K-map of the function below? Note that *X* denotes don't-care values. [2 marks]

$$F6a(A, B, C, D) = \Sigma m(3, 4, 5, 7, 11, 14) + \Sigma X(2, 6, 8, 12, 15)$$

   (b) Given the following magnitude comparator, write the Boolean function *F6b*(*A*,*B*,*C*,*D*) in Σ*m* form. [3 marks]



7. [6 marks]
   The following sequential circuit cycles through 6 states *ABC*. Two states are unused. The circuit is implemented using a *JK* flip-flop for *A*, a *D* flip-flop for *B*, and a *T* flip-flop for *C*.



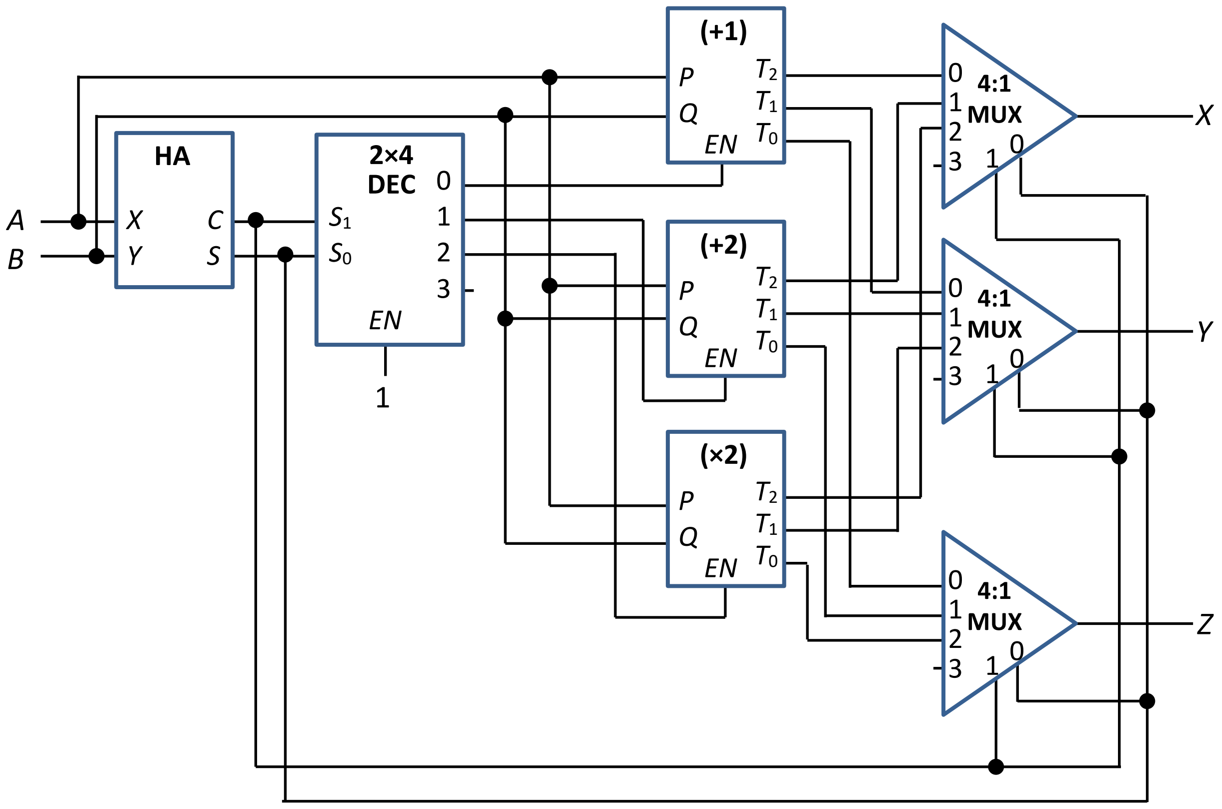
   On the answer sheet, complete the state diagram by filling in the state numbers which are in decimal, as well as the state transition from the unused state. Two states (000 or 0 in decimal, and 110 or 6 in decimal) have been filled for you.

8. [5 marks]
Implement the following function using a single 4:1 multiplexer with at most one additional logic gate. Complemented literals are not available.

$$F8(A, B, C, D) = \Pi M(0, 1, 2, 5, 9, 13)$$

9. [8 marks]
Study the following circuit which uses a half adder, a 2×4 decoder with 1-enable and active high outputs, and three devices each with a 1-enable control (*EN*):

- A (+1)-device: it takes in two inputs *P* and *Q* and produces 3-bit output with value *P*+*Q*+1.
- A (+2)-device: it takes in two inputs *P* and *Q* and produces 3-bit output with value *P*+*Q*+2.
- A (×2)-device: it takes in two inputs *P* and *Q* and produces 3-bit output with value (*P*+*Q*)×2.



The above circuit is too complex. Redesign the circuit using the minimum number of logic gates. Write your expressions for *X*, *Y* and *Z* (6 marks) and draw the logic diagram of your circuit (2 marks). Complemented literals are not available.

10. [6 marks]

Study the MIPS code below, given that **$s0** and **$s1** have been assigned the starting addresses of integer arrays **A** and **B** respectively. You need to determine the value of **$s2** after the code is executed.

The MIPS Reference Data sheet is provided on page 9.

```
            addi $s2, $zero, 0
            addi $t0, $s0, 0
            addi $t1, $s1, 0
            addi $t4, $zero, 0
            addi $t5, $zero, 24

loop:       beq  $t4, $t5, out
            lw   $t2, 0($t0)
            lw   $t3, 0($t1)
            beq  $t2, $zero, out
            beq  $t3, $zero, out

            bne  $t2, $t3, notequal
            add  $s2, $s2, $t2

notequal:   addi $t4, $t4, 4
            add  $t0, $s0, $t4
            add  $t1, $s1, $t4
            j    loop

out:
```

What is the value of **$s2** after the code is executed, given the following arrays?

(a) A = { 1, 2, 3, 4, 5, 0 }
B = { 1, 2, 3, 4, 5, 0 } [2 marks]

(b) A = { -9, -3, -5, 2, -4, 2 }
B = { -6, -3, -7, 0, -4, -8 } [2 marks]

(c) A = { 3, -2, 7, 5, 9, -4, 6, 6, -3, 0, 7, 0 }
B = { 5, -1, 7, -3, 9, -4, 6, 0, 3, 8, 7, 0 } [2 marks]

——— END OF PAPER ———

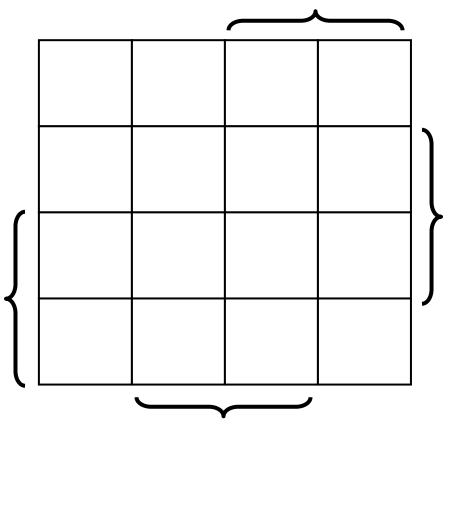
(Blank truth tables, K-maps and state table are provided in the next two pages.)

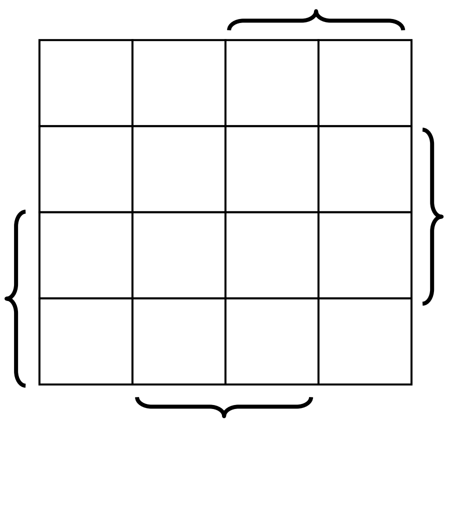
This page is for your rough work.

A	*B*	*C*	*D*	
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

A	*B*	*C*	*D*	
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

A	*B*	*C*	*D*					
0	0	0	0					
0	0	0	1					
0	0	1	0					
0	0	1	1					
0	1	0	0					
0	1	0	1					
0	1	1	0					
0	1	1	1					
1	0	0	0					
1	0	0	1					
1	0	1	0					
1	0	1	1					
1	1	0	0					
1	1	0	1					
1	1	1	0					
1	1	1	1					





This page is for your rough work.

A	*B*	*C*							
0	0	0							
0	0	1							
0	1	0							
0	1	1							
1	0	0							
1	0	1							
1	1	0							
1	1	1							

MIPS Reference Data

①

CORE INSTRUCTION SET

NAME, MNEMONIC		FOR-MAT	OPERATION (in Verilog)		OPCODE / FUNCT (Hex)
Add	`add`	R	R[rd] = R[rs] + R[rt]	(1)	$0 / 20_{hex}$
Add Immediate	`addi`	I	R[rt] = R[rs] + SignExtImm	(1,2)	8_{hex}
Add Imm. Unsigned	`addiu`	I	R[rt] = R[rs] + SignExtImm	(2)	9_{hex}
Add Unsigned	`addu`	R	R[rd] = R[rs] + R[rt]		$0 / 21_{hex}$
And	`and`	R	R[rd] = R[rs] & R[rt]		$0 / 24_{hex}$
And Immediate	`andi`	I	R[rt] = R[rs] & ZeroExtImm	(3)	c_{hex}
Branch On Equal	`beq`	I	if(R[rs]==R[rt]) PC=PC+4+BranchAddr	(4)	4_{hex}
Branch On Not Equal	`bne`	I	if(R[rs]!=R[rt]) PC=PC+4+BranchAddr	(4)	5_{hex}
Jump	`j`	J	PC=JumpAddr	(5)	2_{hex}
Jump And Link	`jal`	J	R[31]=PC+8;PC=JumpAddr	(5)	3_{hex}
Jump Register	`jr`	R	PC=R[rs]		$0 / 08_{hex}$
Load Byte Unsigned	`lbu`	I	R[rt]={24'b0,M[R[rs] +SignExtImm](7:0)}	(2)	24_{hex}
Load Halfword Unsigned	`lhu`	I	R[rt]={16'b0,M[R[rs] +SignExtImm](15:0)}	(2)	25_{hex}
Load Linked	`ll`	I	R[rt] = M[R[rs]+SignExtImm]	(2,7)	30_{hex}
Load Upper Imm.	`lui`	I	R[rt] = {imm, 16'b0}		f_{hex}
Load Word	`lw`	I	R[rt] = M[R[rs]+SignExtImm]	(2)	23_{hex}
Nor	`nor`	R	R[rd] = ~ (R[rs] | R[rt])		$0 / 27_{hex}$
Or	`or`	R	R[rd] = R[rs] | R[rt]		$0 / 25_{hex}$
Or Immediate	`ori`	I	R[rt] = R[rs] | ZeroExtImm	(3)	d_{hex}
Set Less Than	`slt`	R	R[rd] = (R[rs] < R[rt]) ? 1 : 0		$0 / 2a_{hex}$
Set Less Than Imm.	`slti`	I	R[rt] = (R[rs] < SignExtImm)? 1 : 0	(2)	a_{hex}
Set Less Than Imm. Unsigned	`sltiu`	I	R[rt] = (R[rs] < SignExtImm) ? 1 : 0	(2,6)	b_{hex}
Set Less Than Unsig.	`sltu`	R	R[rd] = (R[rs] < R[rt]) ? 1 : 0	(6)	$0 / 2b_{hex}$
Shift Left Logical	`sll`	R	R[rd] = R[rt] << shamt		$0 / 00_{hex}$
Shift Right Logical	`srl`	R	R[rd] = R[rt] >> shamt		$0 / 02_{hex}$
Store Byte	`sb`	I	M[R[rs]+SignExtImm](7:0) = R[rt](7:0)	(2)	28_{hex}
Store Conditional	`sc`	I	M[R[rs]+SignExtImm] = R[rt]; R[rt] = (*atomic*) ? 1 : 0	(2,7)	38_{hex}
Store Halfword	`sh`	I	M[R[rs]+SignExtImm](15:0) = R[rt](15:0)	(2)	29_{hex}
Store Word	`sw`	I	M[R[rs]+SignExtImm] = R[rt]	(2)	$2b_{hex}$
Subtract	`sub`	R	R[rd] = R[rs] - R[rt]	(1)	$0 / 22_{hex}$
Subtract Unsigned	`subu`	R	R[rd] = R[rs] - R[rt]		$0 / 23_{hex}$

(1) May cause overflow exception
(2) SignExtImm = { 16{immediate[15]}, immediate }
(3) ZeroExtImm = { 16{1b'0}, immediate }
(4) BranchAddr = { 14{immediate[15]}, immediate, 2'b0 }
(5) JumpAddr = { PC+4[31:28], address, 2'b0 }
(6) Operands considered unsigned numbers (vs. 2's comp.)
(7) Atomic test&set pair; R[rt] = 1 if pair atomic, 0 if not atomic

BASIC INSTRUCTION FORMATS

R	opcode (31–26)	rs (25–21)	rt (20–16)	rd (15–11)	shamt (10–6)	funct (5–0)
I	opcode (31–26)	rs (25–21)	rt (20–16)	immediate (15–0)		
J	opcode (31–26)	address (25–0)				

ARITHMETIC CORE INSTRUCTION SET

②

NAME, MNEMONIC		FOR-MAT	OPERATION		OPCODE / FMT / FT / FUNCT (Hex)
Branch On FP True	`bc1t`	FI	if(FPcond)PC=PC+4+BranchAddr	(4)	11/8/1/--
Branch On FP False	`bc1f`	FI	if(!FPcond)PC=PC+4+BranchAddr	(4)	11/8/0/--
Divide	`div`	R	Lo=R[rs]/R[rt]; Hi=R[rs]%R[rt]		0/--/--/1a
Divide Unsigned	`divu`	R	Lo=R[rs]/R[rt]; Hi=R[rs]%R[rt]	(6)	0/--/--/1b
FP Add Single	`add.s`	FR	F[fd]= F[fs] + F[ft]		11/10/--/0
FP Add Double	`add.d`	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} + {F[ft],F[ft+1]}		11/11/--/0
FP Compare Single	`c.x.s*`	FR	FPcond = (F[fs] *op* F[ft]) ? 1 : 0		11/10/--/*y*
FP Compare Double	`c.x.d*`	FR	FPcond = ({F[fs],F[fs+1]} *op* {F[ft],F[ft+1]}) ? 1 : 0		11/11/--/*y*
FP Divide Single	`div.s`	FR	F[fd] = F[fs] / F[ft]		11/10/--/3
FP Divide Double	`div.d`	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} / {F[ft],F[ft+1]}		11/11/--/3
FP Multiply Single	`mul.s`	FR	F[fd] = F[fs] * F[ft]		11/10/--/2
FP Multiply Double	`mul.d`	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} * {F[ft],F[ft+1]}		11/11/--/2
FP Subtract Single	`sub.s`	FR	F[fd]=F[fs] - F[ft]		11/10/--/1
FP Subtract Double	`sub.d`	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} - {F[ft],F[ft+1]}		11/11/--/1
Load FP Single	`lwc1`	I	F[rt]=M[R[rs]+SignExtImm]	(2)	31/--/--/--
Load FP Double	`ldc1`	I	F[rt]=M[R[rs]+SignExtImm]; F[rt+1]=M[R[rs]+SignExtImm+4]	(2)	35/--/--/--
Move From Hi	`mfhi`	R	R[rd] = Hi		0 /--/--/10
Move From Lo	`mflo`	R	R[rd] = Lo		0 /--/--/12
Move From Control	`mfc0`	R	R[rd] = CR[rs]		10 /0/--/0
Multiply	`mult`	R	{Hi,Lo} = R[rs] * R[rt]		0/--/--/18
Multiply Unsigned	`multu`	R	{Hi,Lo} = R[rs] * R[rt]	(6)	0/--/--/19
Shift Right Arith.	`sra`	R	R[rd] = R[rt] >>> shamt		0/--/--/3
Store FP Single	`swc1`	I	M[R[rs]+SignExtImm] = F[rt]	(2)	39/--/--/--
Store FP Double	`sdc1`	I	M[R[rs]+SignExtImm] = F[rt]; M[R[rs]+SignExtImm+4] = F[rt+1]	(2)	3d/--/--/--

* (*x* is `eq`, `lt`, or `le`) (*op* is ==, <, or <=) (*y* is 32, 3c, or 3e)

FLOATING-POINT INSTRUCTION FORMATS

FR	opcode (31–26)	fmt (25–21)	ft (20–16)	fs (15–11)	fd (10–6)	funct (5–0)
FI	opcode (31–26)	fmt (25–21)	ft (20–16)	immediate (15–0)		

PSEUDOINSTRUCTION SET

NAME	MNEMONIC	OPERATION
Branch Less Than	`blt`	if(R[rs]<R[rt]) PC = Label
Branch Greater Than	`bgt`	if(R[rs]>R[rt]) PC = Label
Branch Less Than or Equal	`ble`	if(R[rs]<=R[rt]) PC = Label
Branch Greater Than or Equal	`bge`	if(R[rs]>=R[rt]) PC = Label
Load Immediate	`li`	R[rd] = immediate
Move	`move`	R[rd] = R[rs]

REGISTER NAME, NUMBER, USE, CALL CONVENTION

NAME	NUMBER	USE	PRESERVED ACROSS A CALL?
$zero	0	The Constant Value 0	N.A.
$at	1	Assembler Temporary	No
$v0-$v1	2-3	Values for Function Results and Expression Evaluation	No
$a0-$a3	4-7	Arguments	No
$t0-$t7	8-15	Temporaries	No
$s0-$s7	16-23	Saved Temporaries	Yes
$t8-$t9	24-25	Temporaries	No
$k0-$k1	26-27	Reserved for OS Kernel	No
$gp	28	Global Pointer	Yes
$sp	29	Stack Pointer	Yes
$fp	30	Frame Pointer	Yes
$ra	31	Return Address	Yes