**SYN Reference Data Sheet**

*By Ian Gallardo, Sivan Garcia*

**INTRODUCTION**

The SYN assembly language is a modified version of MIPS. Each instruction is 32 bits, with 5 bit opcodes, 5 bit registers, and 22 bit immediate. There are 21 total instructions, with 11 basic instructions (instructions already in MIPS) and 10 unique instructions (instructions not in MIPS). The binary encodings for all instructions and registers have been changed. All instruction names are synonyms of a real instruction or functionality.**INSTRUCTION SET**

|  |  |  |
| --- | --- | --- |
| **NAME** | **FORMAT** | **OPERATION** |
| BASIC | | |
| PLUS | PLUS   $g1, $g2, $g3 | $g1 = $g2 + $g3 |
| MINUS | MINUS  $g1, $g2, $g3 | $g1 = $g2 - $g3 |
| STAR | STAR   $g1, $g2, $g3 | $g1 = $g2 \* $g3 |
| CMP | CMP   $g1, $g2, $g3 | If ($g2 == $g3) $g1 =1 |
| HOP id Equal | HOPEQ $g1, $g2, LABEL | if ($g1 == $g2) goto LABEL |
| HOP If greater than | HOPEGT $g1, $g2, LABEL | if ($g1 > $g2) goto LABEL |
| HOP If less than | HOPELT $g1, $g2, LABEL | if ($g1 < $g2) goto LABEL |
| LEAP | LEAP LABEL | goto LABEL |
| Set Byte | SETB $g1, c | $g1 = c |
| Set Word | SETW $g1, c | $g1 = c |
| Set String | SETS $g1, STR | $g1 = STR |
| UNIQUE | | |
| MOD | MOD $g1, $g2, $g3 | $g1 = $g2 % $g3 |
| Print String | PRINTS $gs | printf(“%s”, $gs) |
| Print Int | PRINTS $gs | printf(“%d”, $gs) |
| Print Float | PRINTS $gs | printf(“%f”, $gs) |
| Read String | READS $gs | scanf(“%s”, $gs) |
| Read Int | READI $gs | scanf(“%d”, $gs) |
| Read Float | READF $gs | scanf(“%f”, $gs) |
| Absolute Value | LOW $g1, $g2 | $g1 = abs($g2) |
| Random Number | RAND $g1, $g2, $g3 | $g1 = rand() % ($g3 - $g2) + $g2 |
| Roll Dice | GAMBLE $g1 | $g1 = rand() % 6 + 1 |
| Plus Immediate | PLUSI $g1, $g2, c | $g1 = $g2 + c |

**BINARY ENCODING**

*s = source register 1*

*t = source register 2*

*d = destination register*

*i = immediate*

*- = don’t care bit*

|  |  |  |
| --- | --- | --- |
| **NAME** | **MNEMONIC** | **BINARY ENCODING** |
| BASIC | | |
| PLUS | PLUS | 00000 ddddd sssss ttttt |
| MINUS | MINUS | 00001 ddddd sssss ttttt |
| STAR | STAR | 00010 ddddd sssss ttttt |
| CMP | CMP | 00011 ddddd sssss |
| HOP if Equal | HOPEQ | 00100 sssss ttttt 1111 1111 iiii iiii iiii iiii |
| HOP If greater than | HOPGT | 00101 sssss ttttt 1111 1111 iiii iiii iiii iiii |
| HOP If less than | HOPLT | 00110 sssss ttttt 1111 1111 iiii iiii iiii iiii |
| LEAP | LEAP | 00111 1111 1111 iiii iiii iiii iiii |
| Set Byte | SETB | 01000 ddddd iiii iiii |
| Set Word | SETW | 01001 ddddd iiii iiii iiii iiii |
| Set String | SETS | 01010 ddddd iiii iiii iiii iiii |
| UNIQUE | | |
| MOD | MOD | 01011 ddddd sssss ttttt |
| Print String | PRINTS | 01100 sssss 10000 |
| Print Int | PRINTI | 01100 sssss 10001 |
| Print Float | PRINTF | 01100 sssss 10010 |
| Read String | READS | 01101 sssss 10011 |
| Read Int | READI | 01101 sssss 10100 |
| Read Float | READF | 01101 sssss 10101 |
| Absolute Value | LOW | 01110 ddddd sssss |
| Random Number | RAND | 01111 ddddd sssss ttttt |
| Roll Dice | GAMBLE | 10000 sssss |
| Plus Immediate | PLUSI | 10001 ddddd sssss iiiii |

**REGISTERS**

|  |  |  |
| --- | --- | --- |
| **REGISTERS** | **PURPOSE** | **BINARY ENCODING** |
| $gz | Always Zero | 00000 |
| $grt | Return value | 00001 |
| $gs | Special register for syscalls | 00010 |
| $ga1 | Arg1 // first argument for functions | 00011 |
| $ga2 | Arg 2 // second argument for functions | 00100 |
| $g1 | Temporary 1 | 10001 |
| $g2 | Temporary 2 | 10010 |
| $g3 | Temporary 3 | 10011 |
| $g4 | Temporary 4 | 10100 |
| $g5 | Temporary 5 | 10101 |
| $g6 | Temporary 6 | 10110 |

**EXAMPLES**

|  |
| --- |
| **ANSWER TO LIFE** |
| C  #include <stdio.h>  void main(){     int val = 0;     int ans = 42;     printf("What is the Answer to life?");     scanf("%d", &val);     if( val == ans){        printf("Right!\n");     }else{        printf("Wrong!\n");     }  } |
| ASSEMBLY  SETB $g1, 0  SETB $g2, 42  SETS $gs, What is the Answer to life?  PRINTS $gs  READI $g1  HOPEQ $g1, $g2, .GOOD  SETS $gs, Wrong!  PRINTS $gs  LEAP .END  .GOOD  SETS $gs, Right!  PRINTS $gs  .END |
| MACHINE CODE (*\* individual instructions have been separated for convenience )*  010001000100000000  010001001000101010  010100001001010111011010000110000101110100  011000001010000  011011000110100  001001000110010111111110010111001000111010011110100111101000100  0101000010010101110111001001101111011011100110011100100001  011000001010000  001111111111100101110010001010100111001000100  111111110010111001000111010011110100111101000100  0101000010010100100110100101100111011010000111010000100001  011000001010000  1111111100101110010001010100111001000100  *\* All label addresses start with tag of 1111 1111 followed by label name* |

|  |
| --- |
| **PRINT DAYS OF THE WEEK** |
| C  #include <stdio.h>  void main(){     printf("Days of the week\n");     printf("Sunday\n");     printf("Monday\n");     printf("Tuesday\n");     printf("Wednesday\n");     printf("Thursday\n");     printf("Friday\n");     printf("Saturday\n");  } |
| ASSEMBLY  SETS $gs, Days of the week  PRINTS $gs  SETS $gs, Sunday  PRINTS $gs  SETS $gs, Monday  PRINTS $gs  SETS $gs, Tuesday  PRINTS $gs  SETS $gs, Wednesday  PRINTS $gs  SETS $gs, Thursday  PRINTS $gs  SETS $gs, Friday  PRINTS $gs  SETS $gs, Saturday  PRINTS $gs |
| MACHINE CODE (*\* individual instructions have been separated for convenience*)  010100001001000100011000010111100101110011  011000001010000  0101000010010100110111010101101110011001000110000101111001  011000001010000  0101000010010011010110111101101110011001000110000101111001  011000001010000  010100001001010100011101010110010101110011011001000110000101111001  011000001010000  0101000010010101110110010101100100011011100110010101110011011001000110000101111001  011000001010000  01010000100101010001101000011101010111001001110011011001000110000101111001  011000001010000  0101000010010001100111001001101001011001000110000101111001  011000001010000  01010000100101001101100001011101000111010101110010011001000110000101111001  011000001010000 |

|  |
| --- |
| **FIZZBUZZ** |
| C  # Performs FizzBuzz over numbers 1 - 100  #include <stdio.h>  int main() {     int i = 1;      while ( i <= 100) {          if (i % 15 == 0) {              printf("FizzBuzz\n");          } else if (i % 3 == 0) {              printf("Fizz\n");          } else if (i % 5 == 0) {              printf("Buzz\n");          } else {              printf("%d\n", i);          }          i = i + 1;      }      return 0;  } |
| ASSEMBLY  SETB $g1, 1  SETB $g2, 100  SETB $g6, 1  .LOOP  HOPEQ $g1, $g2, .END  PLUS $g1, $g1, $g6  SETB $g3, 15  MOD $g4, $g1, $g3  HOPEQ $g4, $gz .FIFTEEN  SETB $g3, 3  MOD $g4, $g1, $g3  HOPEQ $g4, $gz .THREE  SETB $g3, 5  MOD $g4, $g1, $g3  HOPEQ $g4, $gz .FIVE  PRINTI $g1  LEAP .LOOP  .FIFTEEN  SETS $gs FizzBuzz  PRINTS $gs  LEAP .LOOP  .THREE  SETS $gs Fizz  PRINTS $gs  LEAP .LOOP  .FIVE  SETS $gs Buzz  PRINTS $gs  LEAP .LOOP  .END |
| MACHINE CODE (*\* individual instructions have been separated for convenience)*  010001000100000001  010001001001100100  010001011000000001  111111110010111001001100010011110100111101010000  0010010001100101111111100101110010001010100111001000100  00000100011000110110  010001001100001111  01011101001000110011  001001010000000111111110010111001000110010010010100011001010100010001010100010101001110  010001001100000011  01011101001000110011  00100101000000011111111001011100101010001001000010100100100010101000101  010001001100000101  01011101001000110011  001001010000000111111110010111001000110010010010101011001000101  011001000110001  00111111111110010111001001100010011110100111101010000  111111110010111001000110010010010100011001010100010001010100010101001110  01010000100100011001101001011110100111101001000010011101010111101001111010  011000001010000  00111111111110010111001001100010011110100111101010000  11111111001011100101010001001000010100100100010101000101  010100001001000110011010010111101001111010  011000001010000  00111111111110010111001001100010011110100111101010000  111111110010111001000110010010010101011001000101  010100001001000010011101010111101001111010  011000001010000  00111111111110010111001001100010011110100111101010000  1111111100101110010001010100111001000100 |