Первый байт		Второй байт	
0001**** регистр = значение	****xxxx xxxx-адрес регистра	Значение 0-255	
0011**** регистр+=значение	****xxxx xxxx-адрес регистра	Значение 0-255	
0010**** регистр1 ?= регистр2	****0000 = регистр2 ****0001 = -регистр2 ****0010 = |регистр2| ****0011 = !регистр2 ****0100 += регистр2 ****0101 += -регистр2 ****0110 += |регистр2| ****0111 += !регистр2 ****1000 -= регистр2 ****1001 -= -регистр2 ****1010 -= |регистр2| ****1011 -= !регистр2 ****1100 = &регистр2 ****1101 = |регистр2 ****1110 -= регистр2 cmp — устанавливает флаги	Xxxx**** адрес регистра 1	****xxxx адрес регистра 2
0011**** motor	****00** не поворачивать колеса ****01** колеса влево ****10** колеса вправо ****11** ****1100 пропищать ******00 остановить мотор ******01 мотор вперед ******10 мотор назад ******11 спец. См 3 ниже: ****0111 Скорость мотора +10 ****1011 Скорость мотора -10	Время движения (0-вечно)	

	****0011 Значение скорости мотора	Значение 0-255
0100**** переход	****000* безусловный ****001* == ****010* != ****011* > ****100* < ****101* >= ****110* <= ****111* зарезервировано	Адрес 0-255(точнее зависит от 0го бита 1го байта)
0101**** работа с модулем	?? надо обдумать ??	
0110**** cmp устанавливает флаги	регистр	значение
Хм, пока лишь 6 команд, 7ой бит вообще незаюзан. Но так сойдет. Мб введу 64 регистров, среди них 16 ронов и 48 специальных. Правда не знаю под что использовать специальные		

Команда		Соответствие
Rx ... операция с регистром	= число	регистр = значение(не забываем представить в допкоде отрицательное значение числа)
	= ?Rx?	регистр1 = регистр2
	+= число	регистр+=значение
	+= ?Rx?	регистр1 += регистр2
	-= число	Регистр-=значение
	-= ?Rx?	Регистр1 -= регистр2
Move({1-255,forever},{forward,back,stop}, {left,right}) команда, управляющая движением. 3-го параметра может не быть — значит, что едем прямо		motor
Speed(...)	число БЕЗ ЗНАКОВ	Motor > Значение скорости мотора

	-10	Motor > Скорость мотора -10
	+10	Motor > Скорость мотора +10
Label(метка)		Метка. Сохраняй их в map<string,int>, а в инт заноси индекс ячейки байткода
Goto(метка)		Переход к метке (ищи их в map-е) и создавай 2 ячейки байткода с безусловным переходом на тот инт в ячейке
If(Rx оператор ...){}	число	регистр+=значение а затем пиши условный переход
	Rx	регистр1 ?= регистр2 > = -регистр2 а затем пиши условный переход

И вот, наконец, я составил таблицу команд машинки. Хотя бы тех команд, которые нам надо будет сдать как 5ю лабу.

Машинка в процессе работы реагирует на события, каждое из которых выполняет определенную программу.

Байткод состоит из 2х частей:
векторов прерываний(событий) и программы
у машинки немножко видов событий:
onInit — при запуске машинки
onProcess – всегда зацикленно
onMillisecond – на каждую миллисекунду

поэтому 16 байт в начале байткода отводится под векторы, каждый из которых хранит адрес программы. Адреса программ должны быть ЧЕТНЫ. Это связано с тем, что всего в программе 512 байт, а каждый вектор хранит значение 0-255, а потому я это значение умножаю на 2.
соответствие событий ячейкам байткода

Событие	Ячейка
onInit	0
onProcess	1
onMillisecond	2

Классическая программа на языке для машинки и ее представление в байткоде

Программа	Адрес	Байткод
		//векторы прерываний
	0	200
	1	17
	2	0
	3	0
….	...	...
onProcess{		
r1 = 64	17	0001.0001(r1 = 64)
r2 = -32	18	0100.0000 (число 64)
r1 += |r2|	19	0001.0010(r2 = -32)
if(r1 != r2){	20	1110.0000 (число -32)
move(forever,forward,left)	21	0010.0110(r1 += |r2|)
speed(255)	22	0001.0010(адреса регистров)
}	23	0010.1111(r1-r2)
r1=0	24	0001.0010(адреса регистров)
}	25	0100.0100(goto если !=)
	26	31
	27	0001.0001(r1=0)
	28	0000.0000(0)
	29	0100.0000(goto)
	30	17
	31	0011.01.10(move forward,left)
	32	0000.0000 (0-forever)
	33	0011.0011(speed)
	34	1111.1111(255)
	35	0100.0000(goto)
	36	27

Поскольку onProcess зацикливается, то в конце ее нужно написать команду goto в байткод