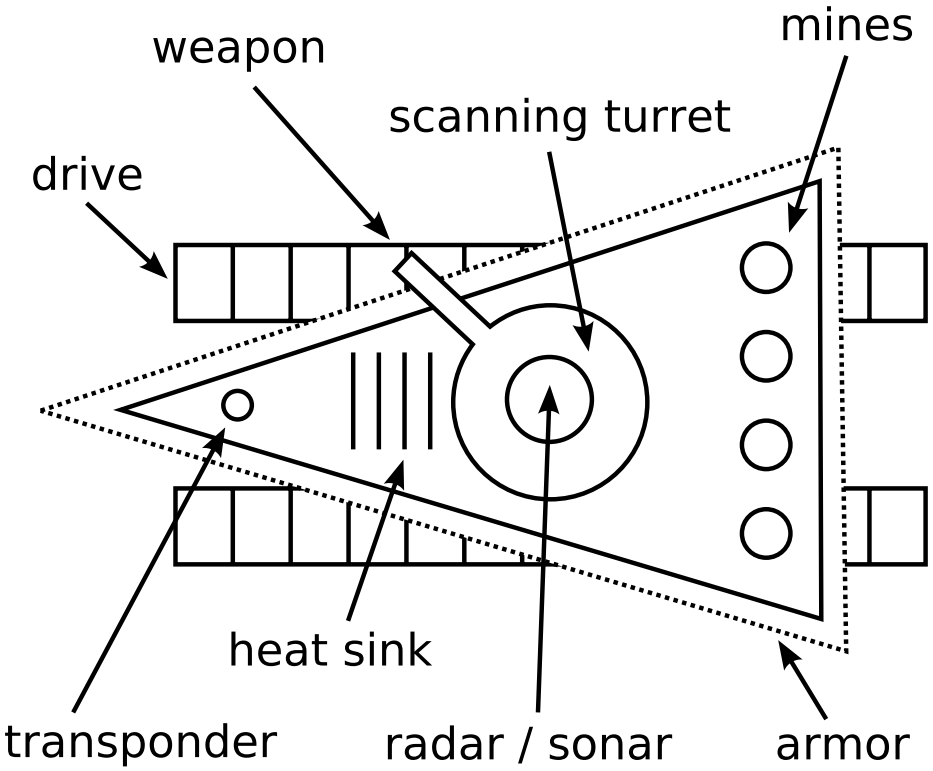
Dissection of an AT-Robot



The Arena

(0,0)

(1000,0)

Armor

Heat

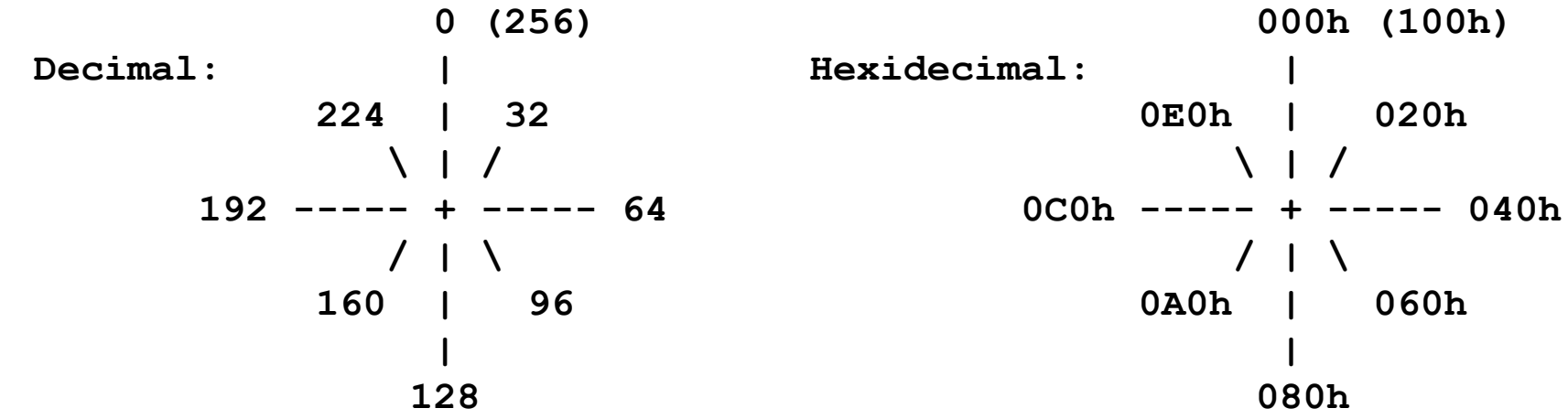
Err Msg

(0,1000)

(1000,1000)

Direction

Navigation is done in a 256 degree circle, with 0 being north. Here are the directions for each course:



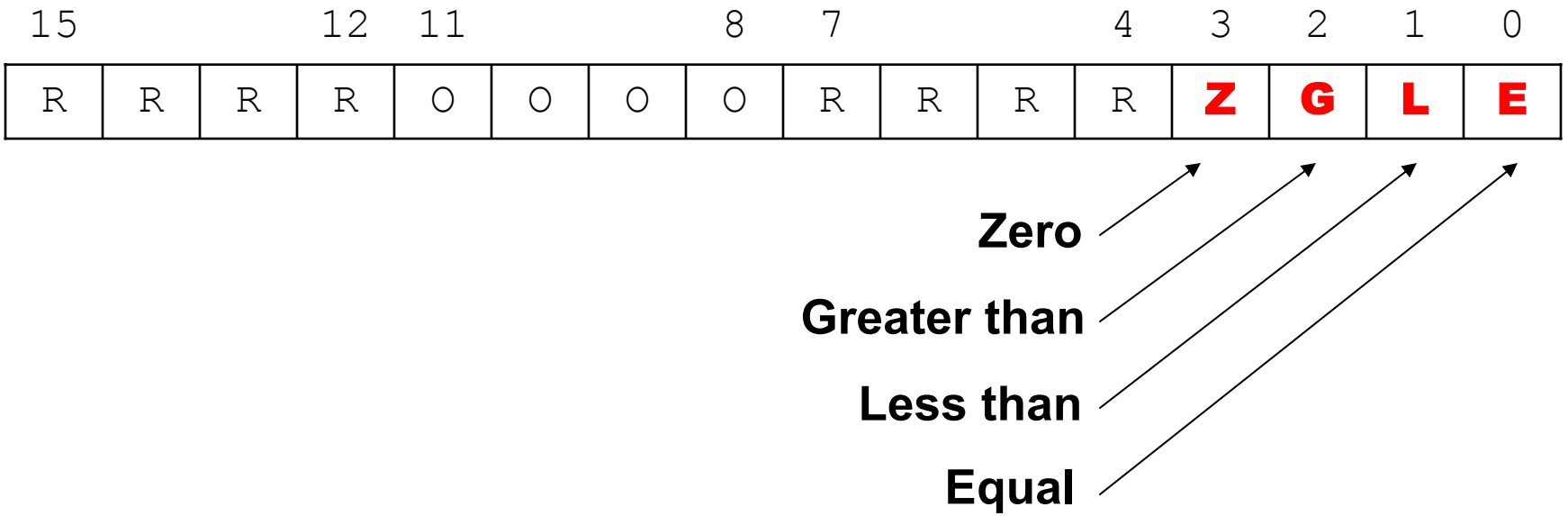
Architecture

- 16-bit signed-only registers *(-32768…32767)*
- Memory is ***word***-addressable

Register	***Name***	***Function***
AX	Accumulator	Interrupt instruction, general use
BX	Base	General use
CX	Counter	Loop control
DX	Data	General use
EX	Extended	Returns from interrupt calls
FX	Function	Returns from interrupt calls
SP	Stack Pointer	Top of stack address
FLAGS	Flags	Comparisons, conditional jumps

Flags Register

- R = reserved
- O = open for developer use

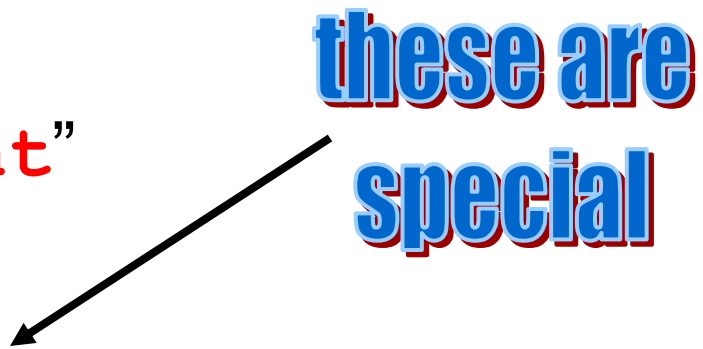


Comparisons

- After a compare instruction (“cmp x, y”)
  - Equal flag: set when x = y
  - Greater flag: set when x > y
  - Less flag: set when x < y
  - Zero flag: set when x = y = 0
- After a test instruction (“test x, y”)
  - Equal flag: set when x = y
  - Zero flag: set when xy = 0 (binary AND)
  - Greater and Less flags: always 0

AT-Robot Statements

- Comments / Remarks
  - "`; this is a comment`"
- Labels (for jumping)
  - "`!my_label_2`" or "`:295`"
- Commands / Instructions
  - "`mov  dx,  64`"
- Variables (only 256 allowed)
  - "`#def  my_variable`"



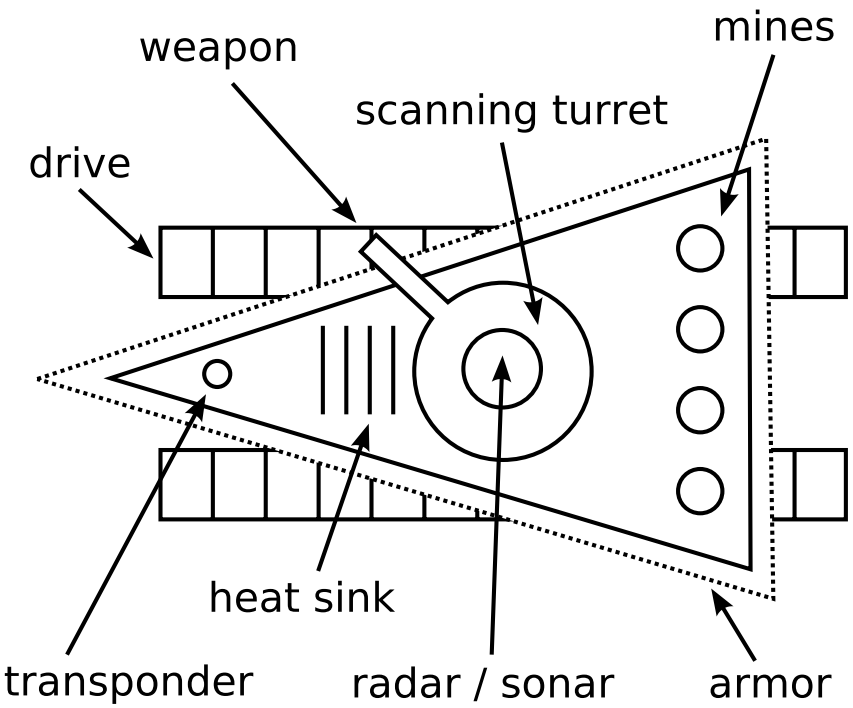
Assembler Directives

Must be declared prior to any statements

- **`#MSG <message>`**
  - displays the **`<message>`** in the robot's display area… 19 characters allowed

- **`#CONFIG <config> = <number>`**
  - assigns a **`<number>`** of points to a particular **`<config>`** option.
  - **`<number>`** is between 0-5

Config Points



- Max of 12 points to distribute toward
  - **scanner**
  - **weapon**
  - **armor**
  - **engine**
  - **heatsinks**
  - **mines**
  - **shield**

Config Effects

Points	Scanner	Weapon	Armor	Engine	Heatsinks	Mines	Shield
0	250	0.50	0.50,1.33	0.50	0.75	2*	None*
1	350	0.80	0.66,1.20	0.80	1.00*	4	-
2	500	1.00*	1.00,1.00*	1.00*	1.125	6	-
3	700	1.20	1.20,0.85	1.20	1.25	10	Weak
4	1000	1.35	1.30,0.75	1.35	1.33	16	Medium
5	1500*	1.50	1.50,0.66	1.50	1.50	24	Strong

Shield damage blocking:

Weak: 2/3 damage gets through, and 2/3 converted to heat. (1/3 overlap)
Medium: 1/2 damage gets through, other half turned to heat.
Strong: 1/3 damage gets through, and 1/3 converted to heat.

Instructions

Time	*Name*	*Syntax / Description*
1	NOP	NOP Simply wastes a clock-cycle.
1	ADD	ADD V N Adds V+N, result stored in V
1	SUB	SUB V N Subtracts V-N, result stored in V
1	INC	INC V Increments V, (v=v+1)
1	DEC	DEC V Decrements V, (v=v-1)
1	SHL	SHL V N Bit-shifts V left N bit positions
1	SHR	SHR V N Bit-shifts V right N bit positions
1	ROL	ROL V N Bit-rotates V left N bit positions
1	ROR	ROR V N Bit-rotates V right N bit positions
1	SAL	SAL V N Bit-Shifts, same as SHL
1	SAR	SAR V N Same as SHR, except preserves bit 15.
1	NEG	NEG V Negates V: V = 0-V (aka "two's compliment")
1	OR	OR V N Bitwise OR, V or N, result stored in V
1	AND	AND V N Bitwise AND, V and N, result stored in V
1	XOR	XOR V N Bitwise XOR, V xor N, result stored in V
1	NOT	NOT V Bitwise NOT, not(V), result stored in V
10	MPY	MPY V N Mulitplies V*N, result stored in V
10	DIV	DIV V N Divides V/N, result stored in V (integer)
10	MOD	MOD V N MOD's V & N, result stored in V (modulus)
1	RET	RET Returns from a subroutine (pops the ip)
1	CALL	CALL N Calls subroutine at label #N (pushes ip)
1	JMP	JMP N Jumps program (ip) to label #N
1	CMP	CMP N N Compares two numbers, results in flags reg.

Instructions

Time	*Name*	*Syntax / Description*	
0	JLS	JLS N	Jumps to label N if last compare was <
0	JGR	JGR N	Jumps to label N if last compare was >
0	JNE	JNE N	Jumps to label N if last compare was <>
0	JE	JEQ N	Jumps to label N if last compare was =
0	JGE	JAE N	Jumps to label N if last compare was >=
0	JLE	JBE N	Jumps to label N if last compare was <=
0	JZ	JZ N	Jumps to label N if last compare was 0
0	JNZ	JNZ N	Jumps to label N if last compare was not 0
1	JTL	JTL N	Jumps to line N of compiled program.
3	XCHG	XCHG V V	Exchanges the values of two variables
1	DO	DO N	Sets CX = N
1	LOOP	LOOP N	Decrements CX, If CX>0 then Jumps to label N
2	TEST	TEST N N	Ands two numbers, result not stored, flags set
1	MOV	MOV V N	Sets V = N
2	ADDR	LOC V V	Sets first V = memory address of second V
2	GET	GET V N	Sets V = number from memory location N
2	PUT	PUT N1 N2	Sets memory location N2 = N1
?	INT	INT N	Executes interrupt number N
4+	IPO/IN	IPO N V	Inputs number from port N, result into V
4+	OPO/OUT	OPO N1 N2	Outputs N2 to port N1
?	DEL	DEL N	Equivalent to N NOPS.
1	PUSH	PUSH N	Puts N onto the stack (sp incremented)
1	POP	POP V	Removes a number from the stack, into V
0	ERR	ERR N	Generate an error code, useful for debugging

Jumping with different label types

```
#def unicorn
   mov unicorn,  12
:12
   mov  ax,  0
   mov  bx,  3
!start
:3
   cmp  ax,  4
   je   !end
   add  ax,  1
   jmp  bx
!end
   jmp  unicorn
```

Ports

Num:	T:	I/O	Name:	Function:
1	0	I	Spedometer	Returns current throttle setting[-75- 100]
2	0	I	Heat Sensor	Returns current heat-level [0 - 500]
3	0	I	Compass	Returns current heading [0 - 255]
4	0	I	Turret Sensor	Returns current turret offset [0 - 255]
5	0	I	Turret Sensor	Returns absolute turret heading [0 - 255]
6	0	I	Damage Sensor	Returns current armor level [0 - 100]
7	1	I	Scanner	Returns range to nearest target in scan arc
8	1	I	Accuracy	Returns accuracy of last scan [-2 - 2]
9	3	I	Radar	Returns range to nearest target
10	0	I	Random Generator	Returns random number [-32768 - 32767]
11	0	O	Throttle	Sets throttle [-75 - 100]
12	0	O	Rotate Turret	Offsets turret (cumulative)

Ports

Num:	T:	I/O	Name:	Function:
13	0	O	Aim Turret	Sets turret offset to value [0 - 255]
14	0	O	Steering	Turn specified number of degrees
15	3	O	Weapon control	Fires weapon w/ angle adjustment [-4 - 4]
16	40	I	Sonar	Returns heading to nearest target[0 - 255]
17	0	I/O	Scan-Arc	Sets/Returns scan-arc width. [0 - 64]
18	0	I/O	Overburn	Sets/Returns overburn status [0 is off, else on]
19	0	I/O	Transponder	Sets/Returns current transponder ID
20	0	I/O	Shutdown-Level	Sets/Returns shutdown-level.
21	0	I/O	Com Channel	Sets/Returns com channel setting
22	0	I/O	Mine Layer	Lays mine or Returns mines-remaining.
23	0	I/O	Mine Trigger	Detonates/returns previously-placed mines.
24	0	I/O	Shield	Sets/Returns shield's status (0=off, else=on)

Interrupts

Num:	T:	Name:	Function:
0	-	Destruct	Detonate the robot (if I go down, you go down with me!)
1	10	Reset	Resets robot program.
2	5	Locate	Sets EX,FX registers equal to X,Y coordinates.
3	2	Keepshift	Sets Keepshift, input: AX 0 = off, non-0 = on
4	1	Overburn	Sets Overburn, input: AX 0 = off, non-0 = on
5	2	ID	Returns robot ID number in FX
6	2	Timer	Returns game clock in EX:FX (32-bit number)
7	32	Find Angle	Returns angle to point specified in EX,FX; AX=result
8	1	Target-ID	Returns ID of last robot scanned (with any scan) in FX
9	2	Target-Info	Returns info on last scanned target (EX=dir,FX=throttle)

Interrupts

```
Num: T: Name:         Function:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 10   4 Game-Info     Returns info: DX=Total number of robots active,
                                    EX=Match number,
                                    FX=Number of matches
 11   5 Robot-info    Returns info: DX=Robot speed (in cm per game-cycle),
                                    EX=Time since last damage taken
                                    FX=Time since a fired shot hit a robot.
                                    (time measured in game-cycles)
                                    (robot speed is current as of int call)
 12   1 Collisions    Returns collision count in FX
 13   1 Reset ColCnt  Resets collision count back to 0.
 14   1 Transmit      Transmits the data in AX on the current channel.
 15   1 Receive       Returns the next item in com queue in FX
 16   1 DataReady     Returns the amount of data in queue in FX (0 for none).
 17   1 ClearCom      Empties the Com Queue
 18   3 Kills/Deaths  Returns info: DX=Kill Count (spans multiple rounds)
                                    EX=Kill Count (for this round only)
                                    FX=Deaths     (spans multiple rounds)
 19   1 ClearMeters   Resets the 'meters' variable to 0.
```

Memory Map

Addr:	Name:	Function:
0	Dspd	Desired speed robot is trying to achieve.
1	Dhd	Desired heading robot is trying to achieve.
2	tpos	Current turret offset
3	acc	accuracy value from last scan
4	swap	temporary swap space used by swap/xchg instruction
5	tr-id	ID of last target scanned (by any scan).
6	tr-dir	Relative heading of last target scanned.
7	tr-spd	Throttle of last target scanned.
8	ColCnt	Collision count.
9	Meters	Meters travelled. 15 bits used.. (32767+1)=0
10	ComBase	Current base of the communications queue
11	ComEnd	Current end-point of the communications queue
13	tr-vel	Absolute speed (cm/cycle) of last target scanned
14-63	res	reserved

Memory Map

Addr:	Name:	Function:
64	flags	flags register
65	ax	ax register
66	bx	bx register
67	cx	cx register
68	dx	dx register
69	ex	ex register
70	fx	fx register
71	sp	stack pointer
72-95	res	reserved for future registers
96-127	res	reserved
128-384	var	variable space
385-511	user	user space
512-767	com-queue	Communications receiving queue
768-1023	res	Reserved space
1024...	code	Robot program (code segment) (ROM)

Constants

Value:	Mnemonic:	Port-Constant:	Interrupt-Constant:	Generic Constant:
-32768				MININT
32767				MAXINT
0	NOP		I_DESTRUCT	
1	ADD	P_SPEDOMETER	I_RESET	
2	SUB	P_HEAT	I_LOCATE	
3	OR	P_COMPASS	I_KEEPSHIFT	
4	AND	P_TURRET_OFS	I_OVERBURN	
5	XOR	P_TURRET_ABS	I_ID	
6	NOT	P_DAMAGE, P_ARMOR	I_TIMER	
7	MPY	P_SCAN	I_ANGLE	
8	DIV	P_ACCURACY	I_TID, I_TARGETID	
9	MOD	P_RADAR	I_TINFO, I_TARGETINFO	
10	RET	P_RANDOM, P_RAND	I_GINFO, I_GAMEINFO	
11	GSB, CALL	P_THROTTLE	I_RINFO, I_ROBOTINFO	
12	JMP, GOTO	P_OFS_TURRET, P_TROTATE	I_COLLISIONS	
13	JLS, JB	P_ABS_TURRET, P_TAIM	I_RESETCOLCNT	
14	JGR, JA	P_STEERING	I_TRANSMIT	
15	JNE	P_WEAP, P_WEAPON,P_FIRE	I_RECEIVE	
16	JEQ, JE	P_SONAR	I_DATAREADY	
17	XCHG, SWAP	P_ARC, P_SCANARC	I_CLEARCOM	
18	DO	P_OVERBURN	I_KILLS, I_DEATHS	
19	LOOP	P_TRANSPONDER	I_CLEARMETERS	
20	CMP	P_SHUTDOWN		
21	TEST	P_CHANNEL		
22	SET, MOV	P_MINELAYER		
23	LOC	P_MINETRIGGER		
24	GET	P_SHIELD,P_SHIELDS		

Constants (cont.)

Value:	Mnemonic:	Port-Constant:	Interrupt-Constant:	Generic Constant:
25	PUT			
26	INT			
27	IPO, IN			
28	OPO, OUT			
29	DEL, DELAY			
30	PUSH			
31	POP			
32	ERR, ERROR			
33	INC			
34	DEC			
35	SHL			
36	SHR			
37	ROL			
38	ROR			
39	JZ			
40	JNZ			
41	JAE, JGE			
42	JBE, JLE			
43	SAL			
44	SAR			
45	NEG			
46	JTL			