Jonathan Huynh

Final Project Module 1

The custom language assembly is going to be based on the game Pokémon, name pending, using relatively simplified mechanics of how a Pokémon battle would go about, based on MIPS. The language would allow for control of Pokémon stats, combat ability, status effects, battling, and some sort of catching mechanic.

Registers: Some registers that will be used for the Pokémon language

Register	Name	Details
$t0	HP	Amount of health before fainting
$t1	ATK	Amount of damage able to do
$t2	DEF	Amount of damage reduction
$t3	OPP HP	
$t4	OPP ATK	
$t5	OPP DEF	
$t6	Status	Different bits for statuses (1=sleep, 2=poison, 4=burn, 8=paralysis)
$t7	OPP Status	
$s0	Weather	Different amounts for weather (0=clear, 1=rainy, 2=sunny)
$s1	Catch	Catch modifier
$s2	Type	Influences multipliers if applicable
$s3	OPP Type	
$zero	constant 0	Doesn’t hurt to have it
$ra	return address	Will inevitably need this

Instruction Set: Instructions that will be used in the custom language

Name	Syntax	Type	Binary Encoding	Purpose
hpup	hpup $tX, imm	I	001000 rs rt imm	Heals by imm amount (heal by item)
hpdown	hpdown $TX, imm	I	001001 rs, rt, imm	Apply dmg to Pokemon
atk	atk $rd, $rs, $rt	R	000000 rs rt rd 00000 000001	Compute raw dmg (atk-def)
appdmg	appdmg $hp, $dmg	R	000000 rs rt rd 00000 000010	Subtract dmg value from hp
heal	heal $hp, $amt	I	001010 rs rt imm	Restore hp using move
atkup	atkup $tX, imm	I	001011 rs rt imm	Increase atk stat
defup	defup $tx, imm	I	001100 rs rt imm	Increase def stat
cpy	cpy $rd, $rs	R	000000 rs 00000 rd 00000 000011	Copy stat/value
faint	faint $hp, label	I	000101 rs rt imm	Branches if HP<=0
bstatus	bstatus $tX, imm, label	I	000110 rs rt imm	Branch is status matches
typeadv	typeadv $dmg, $aType, $dType	R	000000 rs rt rd 00000 010000	Apply type multiplier
stab	stab $dmg, $mType, $pType	R	000000 rs rt rd 00000 010001	Apply Same-Type Attack Bonus (STAB)
crit	crit $dmg	R	000000 rs 00000 rs 00000 010010	Double damage
statset	statset $status, imm	I	001101 rs rt imm	Inflicts condition
statclr	statclr $status, imm	I	001110 rs rt imm	Remove selected status bit

weather	weather $s0, imm	I	001111 rs rt imm	Set environment
switch	switch $tA, $tB	R	000000 rs rt 00000 00000 010011	Swap stats between registers (switching Pokémon)
gheal	gheal $tX	R	000000 rs 00000 00000 00000 010100	Restore X hp to the entire team
catch	catch $chance, $hp	R	000000 rs rt rd 00000 010101	Compute catch
trainer	trainer label	J	010000 address	Display Trainer

Example Programs

Basic Attack

```
# compute + apply basic dmg
atk $t2, $t1, $t5      # dmg = atk - def
appdmg $t3, $t2     # opp hp -= dmg
faint $t3, opp_faint

j battle_cont

opp_faint:
trainer victory
battle_cont:
```

Type Move w/ STAB and Type Advantage

```
# type (fire) attack with STAB
stab $t2, $s4, $s2          # STAB is fire-type
typeadv $t2, $s4, $s3     # apply type effectiveness
crit $t2                        # crit hit
appdmg $t3, $t2           # apply dmg
```

Healing/Status Clearing

```
hpup $t0, 50                # item restores 50 hp
```

```
statclr $t6, XXX        # curing status effect (sleep)
weather $s0, 2          # set weather → sunny
```

Overall, this custom ASM, while not the most original idea, provides a relatively (to me) unique take on MIPS using the mechanics of a Pokémon battle.